НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Система віддаленого управління автоматизованою рухомою платформою»

Виконала:
студентка IV курсу, групи ІТ-51
Цитовцева Анна Сергіївна _____

Керівник:
професор, к.т.н., проф. Шемаєв Володимир Миколайович _____

Рецензент:
Директор інституту інформаційних технологій в економіці,
к.е.н. Ващаєв Сергій Сергійович _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 рік

Пояснювальна записка
до дипломного проекту
на тему: «Система віддаленого управління автоматизованою рухомою платформою»

Київ – 2019 рік

АНОТАЦІЯ

Цитовцева А.С. Система віддаленого управління автоматизованої рухомої платформи. КПІ ім. Ігоря Сікорського, Київ, 2019.

Проект містить 78 с. тексту, 36 рисунків, 4 таблиці, посилання на 17 літературних джерела, та 5 додатків.

Ключові слова: автоматизована рухома платформа, впровадження залежностей, медіа-файл, особистий кабінет користувача, система управління, Java, Postgres, Raspberry PI.

Об’єктом розробки є система віддаленого управління автоматизованої рухомої платформи.

Метою розробки є програмно-апаратна інфраструктура, яка дозволяє керувати траєкторією руху об’єкта управління з прямою трансляцією відео, а також надає можливість створювати та зберігати медіа-файли.

У дипломному проекті було розроблено систему управління, яка включає такі рівні абстракції як клієнтський, рівень бізнес логіки, рівень взаємодії з базою даних та фізичний рівень. Значну увагу було приділено розробці універсального інтерфейсу управління автоматизованою рухомою платформою та його конкретній реалізації. Фізична частина побудована на основі мікрокомп’ютера Raspberry PI з General Purpose Input/Output для керування периферійними пристроями, на якому встановлена Linux система – дистрибутив Raspbian. В процесі розробки була використана мова програмування Java та фреймворк Spring Boot, система управління базами даних Postrges та середовище розробки Intellij Idea. Для стилізація використано CSS фреймворк Bootstrap.

SUMMARY

Tsytovtseva A.S. Remote control system for automated mobile platform. Igor Sikorsky KPI, Kyiv, 2019.

The project contains 78 s. text, 36 figures, 4 tables and links to 17 literary sources.

Keywords: automated moving platform, dependency implementation, media file, user's personal cabinet, control system, Java, Postgres, Raspberry PI.

The object of development is a remote control of an automated moving platform.

The purpose of the development is hardware and software infrastructure that allows you to control the motion trajectory of the control object with live streaming video, and also provides the ability to create and save media files.

In the diploma project, a management system was developed that includes such levels of abstraction as client, level of business logic, level of interaction with the database and applied level. Considerable attention was paid to the development of a universal control interface for an automated mobile platform and its specific implementation. The physical part is based on the Raspberry PI microprocessor from General Purpose Input / Output to control peripherals, which has the Linux system - the Armbian distribution. In the development process, the Java programming language and the Spring Boot framework, the Postrges database management system and the Intellij Idea development environment were used. For styling, CSS framework Bootstrap is used.

ЗМІСТ

ВСТУП

Актуальність теми

У даній роботі розроблена система віддаленого управління автоматизованої рухомої платформи(СВУАРП). Вона призначена для віддаленого дослідження території без присутності людини на платформі. Архітектура СВУАРП містить апаратну і програмну частини. Апаратна частина дуже подібна на будову людини: датчики відповідають за функції органів чуття, виконавчі пристрої виконують функції рук та ніг. За роль мозку відповідає мікрокомп’ютер, що побудований на базі Raspberry Pi 3. З огляду на таку аналогію архітектура СВУАРП може бути автоматичною, автоматизованої і в перспективі містити елементи штучного інтелекту.

Під час розробки СВУАРП потрібно було вирішити багато практичних питань, які охоплюють аспекти комерційної ефективності, технологічності, стандартизації, безпеки, точності, надійності, сумісність, технічного супроводження і так далі. Саме ці аспекти будуть проаналізовані в даному дипломному проекті.

Мета і задачі дослідження

Мета цієї роботи – дослідити способи управління рухомими платформами, розробити кабінет користувача, який буде включати інтерфейс управління об’єктом та забезпечить високу і незалежну доступність створених медіа файлів та основі прав доступу користувача.

Об’єкт і предмет дослідження

Об'єктом дослідження є процес віддаленого управління рухомою платформою. Предмет дослідження – модель програмно-апаратної інфраструктури СВУАРП.

Методи дослідження

Для досягнення поставленої мети необхідно:

- побудувати прототип;
- дослідити можливе використання алгоритмів управління;

- побудувати систему передачі поточного відео с рухомої системи;
- дослідити алгоритми та програмні засоби збереження медіа файлів;
- провести аналіз методів побудови існуючих веб-сервісів віддаленого управління;
- сформулювати вимоги до безпечного збереження даних;
- розробити структура бази даних для зберігання інформації;
- розробити автоматизовану систему управління.

Практичне значення наукових результатів

Діапазон використання автоматизованих рухомих об'єктів дуже широкий. Розроблену СВУАРП можна використовувати при подальшій розробці системи, що спрямовані на віддалене дослідження території об’єктів промисловості будівель тощо. СВУАРП також має забезпечити збереження даних та управління життєвим циклом інформації.

Апробація результатів роботи

Конференція

Структура роботи

У розділі 1 представлено дослідження автоматизованих систем управління та апартатних інфраструктур, а також їх переваги і недоліки.

У розділі 2 розроблені та описані технічні та функціональні вимоги до системи.

У розділі 3 описано процес побудови прототипу рухомої плаформи і компоненти необхідні для його побудови.

У розділі 4 описані тіхнології та інструменти, які були використані для розробки серверної частини.

У розділі 5 описан процес розробки та основні алгоритми взаємодії з системою.

У розділі 6 була протестована система.

1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Розвиток сучасних технологій дає можливість здійснити автоматизацію процесів управління та сбір інформації з використанням сучасних мов програмування(у тому числі Java) та реалізувати СВУАРП за відносно невеликий час та відносно невеликими витратами. Проте при виборі з усіх запропонованих рішень дуже важливо приділяти увагу не тільки одному певному аспекту, а комплексу функцій, які надає ця система. У цьому розділі були проаналізовані практичні рішення щодо рухомих платформ, які мають невелику вартість, достатній функціонал для реалізації прототипу СВУАРП.

1.1 Аналіз існуючих платформ управління

Серед запропонованих на ринку товарів можна виділити дві категорії рухомих платформ. А саме ті, що керуються через мобільні додатки і передають інформацію на пульт дистанційного керування (ПДК) і такі, що можуть передавати інформацію на ПДК і одночасно зберігати її автономно на борту базового мікрокомп’ютера[1-20]. Проведемо аналіз за актуальними даними у мережі інтернет.

Прикладом представника першої категорії було обрано платформу I-Spy Mini, зображену на рисунку 1.1[10]. Вона обладнана вбудованою відеокамерою, яка передає потокове відео і має наступні характеристики:

- час роботи 15-25 хв;
- час зарядки 120 хв;
- радіус дії 15-30 м;
- максимальна швидкість 3-5 км / год;
- елемент живлення вбудований літій-полімерний 1S акумулятор;
- управління Android / IOS;
- розміри 12 x 9.7 x 5.8 см;
- вага 0.170 кг;

– матеріал пластик;
– камера 640x480, 30 кадр / сек.

Для управління використовується iOS або Android застосунок, що підключається до нього через точку доступу Wi-Fi. Тобто, вимогою для використання є наявність мобільного пристрою, що, на мій погляд, робить його не придатним для використання в умовах підприємства, де трансляція зображення має бути на робочій станції та на більшому екрані. Також процес імпорту великої кількості медіа файлів з використанням мобільного пристрою не є зручним.

Рисунок 1.1 – I-Spy Mini

Представником другої категорії було обрано радіокеровану баггі, зображену на рисунку 1.2 [11]. Її технічні характеристики схожі на характеристики першого представника, проте відрізняються способом збереження інформації. Нижче подано характеристики пристрою:

– час роботи 15 хв;
– баггі розганяється до 20 км / год;

- накопичувач Micro SD 8 Гб;
- радіус дії близько 50 м;
- розміри баггі 30x19x16 см;
- повний привід 4WD;
- акумулятор 4,8 В 500 мАг Ni-cd;
- трансляція відео на смартфон в реальному часі;
- відмінне зчеплення з будь-якою поверхнею;
- подолання підйомів з кутом до 45 градусів.

Рисунок 1.2 – Радіокерована баггі SuboTech RTR 2.4GHz WiFi

Медіа файли зберігаються на самому пристрої, здебільшого на карті пам’яті Micro SD. Тому у випадку, якщо пристрій буде загублено або карта пам’яті загублена, дані не можна буде відновити. У випадках, коли медіа файли мали велику цінність або неможлива ситуація, коли можна отримати такі ж самі данні, наслідками можуть бути великі збитки.

Проаналізувавши представлені рішення, можна виділити ряд основних недоліків, що можуть завадити застосовувати розглянуті рішення в якості прототипу СВУАРП:

- для управління потрібен спеціальний застосунок;
- надійне збереження даних відсутнє;
- деякі пристрої вимагають прокладки кабельних мереж для інтеграції в існуючу структуру;
- ринкова вартість готового рішення в декілька разів дорожча, навіть в базовій комплектації.

1.2 Аналіз комплектів апаратної інфраструктури

На теперішній час для спрощення етапу розробки і комплектації прототипу СВУАРП деякі компанії, наприклад Dexter Industries, пропонують готові програмно-апаратні комплекти [12]. Вони включають в себе основні елементи для побудови рухомої платформи на основі мікрокомп'ютера Raspberry Pi. Користувач може обрати базову комплектацію, проте також доступні варіанти з розширеними можливостями отримання даних з середовища.

Нижче наведено роботехнічні комплекти від Dexter Industries, які використовується для побудови рухомих платформ:

- BrickPi;
- GoPiGo;
- PivotPi.

Розглянемо запропоновані апаратні і програмні інфраструктури в якості потенційного прототипу реалізації СВУАРП.

1.2.1 Апаратна інфраструктура BrickPi

Першим прототипом апаратної інфраструктури СВУАРП може бути BrickPi [12]. BrickPi підходить для тих платформ, які вже мають датчики та двигуни Lego Mindstorms. BrickPi підключається до Raspberry Pi і замінює Mindstorms Brick (мозок системи MINDSTORMS). На рисунку 1.3 представлене поєднання комплекту BrickPi з деталями Lego. Lego Mindstorms є популярною і потужною

робототехнічною платформою, але при вивченні розвитку для мережевого світу існують деякі серйозні обмеження. Система BrickPi побудована щоб додати функціональність і забезпечити можливість підключення до веб-сервісів, об'єднавши декількох роботів разом. BrickPi працює через Wi-Fi та з усіма датчиками та двигунами NXT та EV3. Плата BrickPi3 у поєднанні з деталями Lego [12]. BrickPi початкова комплектація включає в себе:

- плата BrickPi3;
- BrickPi 6-сторонній акриловий корпус;
- блок живлення від батареї (батареї 8AA не включені);
- карта памяті microSD, але для роботи потрібна програма Raspbian, яку потрібно завантажити і встановити самостійно;
- Raspberry Pi 3;
- кабель Ethernet;
- світловий та кольоровий датчик.

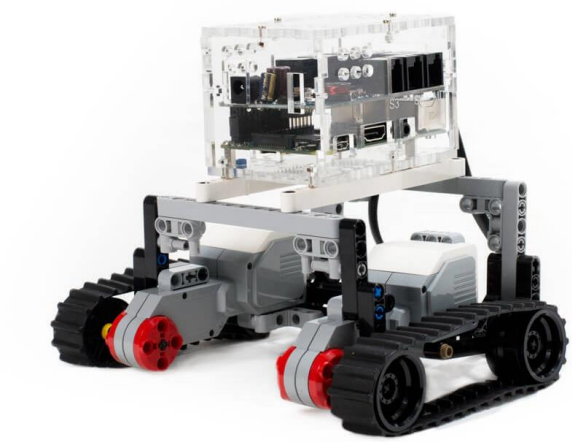

Рисунок 1.3 – Плата BrickPi3 у поєднанні з деталями Lego

Така збірка апаратної інфраструктури передбачає поєднання з технічними частинами Lego, проте якщо метою є придбання повність зібраної платформи, то є сенс надати перевагу іншим комплектам. Початковий комплект BrickPi3 коштує $179.99.

1.2.2 Апаратна інфраструктура GoPiGo

Другим прототипом апаратної інфраструктури СВУАРП може бути GoPiGo [13]. Платформа GoPiGo – це роботехнічний автомобіль. Базовий комплект складається з усіх необхідних компонентів, потрібних для приведення до руху роботехнічного автомобіля. Такий комплект включає в себе:

- плату GoPiGo3;
- Raspberry Pi 3;
- пакет сервісу GoPiGo;
- датчик відстані;
- карту microSD (з програмним забезпеченням DexterOS):
- USB-накопичувач 8 Гб;
- блок живлення.

DexterOS, програмне забезпечення, що постачається з GoPiGo Starter Kit, має десятки безкоштовних вбудованих уроків щоб навчити програмувати GoPiGo в Bloxter (мова програмування, розроблена компанією Dexter, схожа на мову Scratch). Комплект GoPiGo3 у зібраному вигляді зображено на рисунку 1.4 [12].

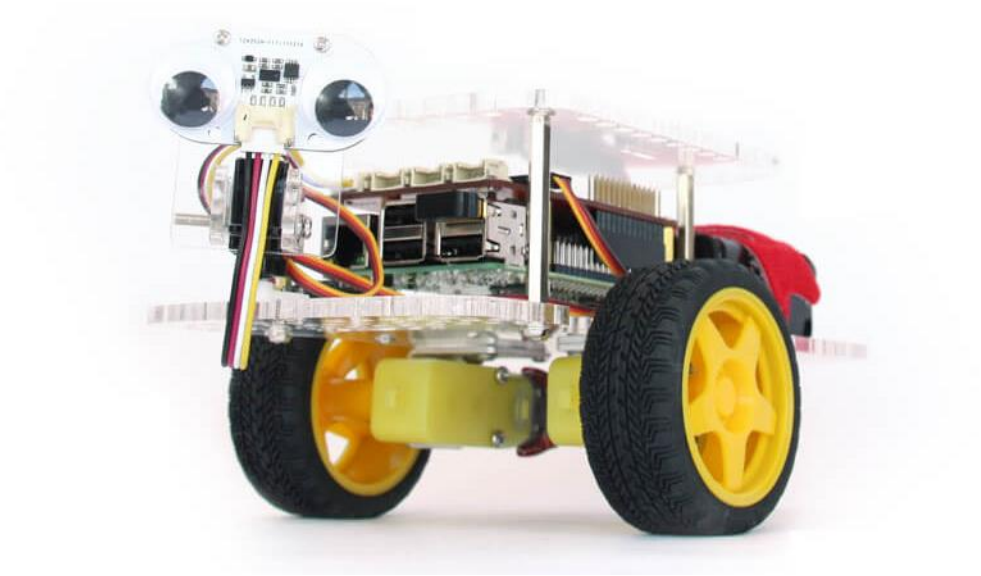

Рисунок 1.4 – Комплект GoPiGo3 у зібраному вигляді

GoPiGo3 стартової комплектації коштує $199.99 і включає в себе усі деталі фізичного рівня. Також особливістю цього комплекту є те, що його програмна частина повністю побудована на основі продуктів компанії, включаючи операційну систему і мову програмування, яка використовується для керування апаратною частиною. Якщо розробник вже має стек технологій, готових до застосування, то доведеться зробити ряд налаштувань для підготовки операційної системи або її переустановлення.

1.2.3 Апаратна інфраструктура PivotPi

Другим прототипом апаратної інфраструктури СВУАРП може бути PivotPi [13]. PivotPi – це повний комплект апаратної інфраструктури від компанії Dexter для перетворення будь-чого у світі на рухомого робота. PivotPi працює на тому рівні складності, на якому ви перебуваєте – підходить і для тривіальної інтеграції STEAM концепції, і для багатоетапної побудови робототехнічної руки. Він працює з усіма версіями Raspberry Pi і представлений на рисунку 1.5 [12].

PivotPi Starter Kit включає в себе все необхідне, щоб розпочати будівництво анімованих роботів і коштує $99.99.

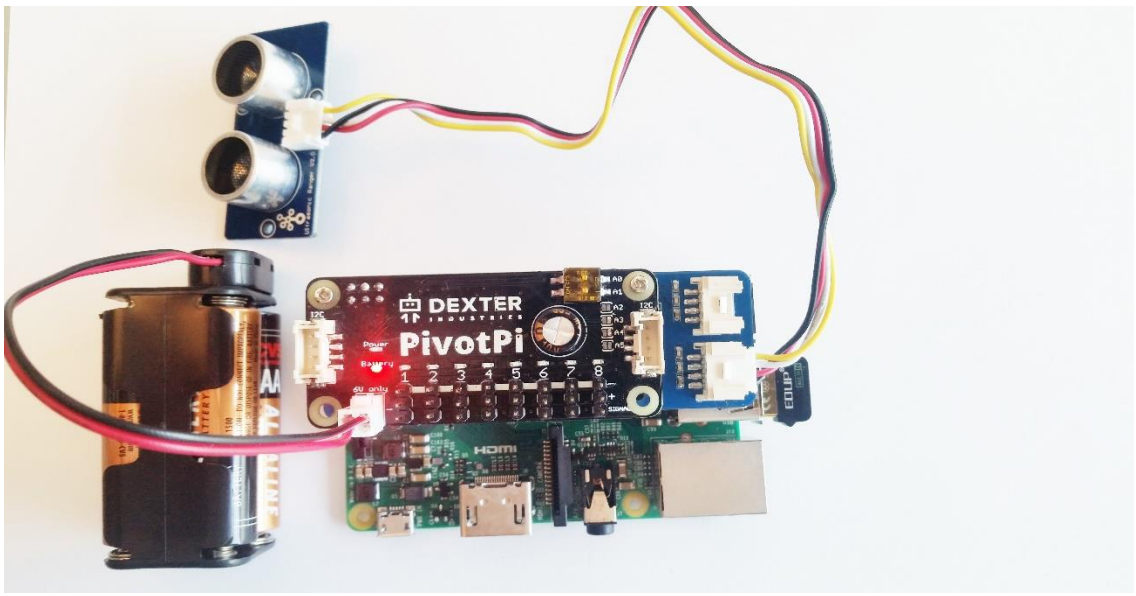

Рисунок 1.5 – Комплектація PivotPi

До складу комплекту PivotPi Starter Kit входить:

- плата Pivot;
- акумуляторну батарею;
- 2 великих серводвигуна і 4 невеликих сервомотора;
- Raspberry Pi;
- кабель Ethernet;
- адаптер живлення;
- Wifi Dongle;
- SD-карту.

Такий набір повністю задовольняє потреби у логічній обробці даних, проте передбачає подальше придбання та інтеграцію з компонентами, для яких було ініційовано управління, наприклад, шасі, сенсори або корпус рухомого об’єкта.

1.3 Висновки до розділу

У цьому розділі були розглянуті сучасні комплексні рішення систем автоматизованого управління та апаратні інфраструктури, що готові до інтеграції з програмним забезпеченням.

Було проведено аналіз щодо використання типових програмно-апаратних комплектів для реалізації СВУАРП. У всіх розглянутих рішеннях визначено один з базових недоліків, що може привести до значних часових і фінансових витрат під час СВУАРП. Таким недоліком є використання спеціалізованої мови програмування для даних платформ. Це може привести до наступних небажаних наслідків:

- необхідність пошуку або підготовки необхідних спеціалістів для даної нерозповсюдженої мови програмування;
- можливі помилки, що знаходяться в компіляторі даної мови програмування; додаткові компілятори – додаткові помилки;
- труднощі з використанням типових бібліотек для розповсюджених мов програмування.

2 РОЗРОБКА ВИМОГ ДО ПРОЕКТУ

Після проведеного аналізу існуючих рішень сформулюємо основні вимоги до інфраструктури СВУАРП, що розроблюється. Першим етапом будуть визначені технічні вимоги, після чого будуть розроблені функціональні вимоги.

2.1 Опис технічних вимог системи

Розроблювана СВУАРП є комплексною автоматизованою системою управління з елементами автоматики. Клієнтський доступ повинен надаватися незалежно від платформи запуску, тому було вирішено організувати взаємодію користувача з рухомою платформою через веб-застосунок.

Важливою вимогою до системи є універсальний доступ до створених медіа-файлів, який не залежить від фізичного носія даних. Тому система повинна зберігати файли до бази даних, які будуть доступні через особистий кабінет користувача.

Оскільки система буде розвиватись і розширювати свої функціональні можливості, апаратна інфраструктура повинна бути побудована на базі потужного мікрокомп’ютера, придатного до підключення нових модулів, датчиків та сенсорів.

Також розширення системи не повинно ускладнювати її супровід та модифікацію компонентів. Така властивість може бути досягнута за допомогою модульності системи, яка дозволить змінювати, переписувати, додавати новий функціонал не впливаючи на все існуючий.

Що стосується безпеки, вимогою є шифрування користувацьких паролів крипостійкими алгоритмами. Також обов’язковим повинне бути приховання усіх чутливих HTTP-відповідей, включаючи секретні токени, дані аутентифікації користувача та сесії. Такі вразливості можуть стати воротами для зловмисників та крадіжки персональних даних.

Наступною вимогою безпеки є перевірка введеного пароля користувача на складність. Система повинна попереджувати про безпечність введеного пароля та забороняти створювати прості паролі для облікового запису користувача.

Також вимогою безпеки є стійкість системи до SQL-ін’єкцій, адже атака на базу даних, яка дозволить виконати деяку дію, яка не планувалося, може призвести до розкриттю даних користувача та поставить під загрозу коректну роботу системи.

Останньою вимогою є впровадження комплексної системи логування різних рівнів, яка дозволить відстежувати проблеми та всю інформацію про них для того щоб відтворити та полагодити.

2.2 Опис функціональних вимог системи

Функціональні вимоги регламентують поведінку додатка в тій чи іншій ситуації, ставлять завдання, які повинна виконувати система.

Вимоги до процесу реєстрації користувача в системі включають такі пункти:

- користувач має можливість зареєструватися в системі за допомогою електронної пошти;
- після реєстрації користувач повинен підтвердити реєстрацію с системі через електронну пошту;
- посилання на підтвердження реєстрації дійсне 24 години;
- користувач має можливість реєструватися через соціальні мережі;
- якщо користувач зареєструвався в системі за допомогою соціальної мережі, підтвердження через електронну пошту не потрібне.

Вимоги до процесу взаємодії користувача з обліковим записом включають такі пункти:

- користувач може входити в систему за допомогою логіну та паролю;
- користувач може входити в систему через соціальну мережу;
- користувач може оновлювати дані власного профіля;

- користувач може змінювати пароль;
- користувач може відновлювати пароль

Функціональні вимоги до процесу управління рухомою платформою включають такі пункти:

- користувач може управляти траєкторією рухомої платформи(повертати вправо-вліво, вперед-назад);
- користувач бачить відео, яке транслюється з камери рухомої платформи;
- користувач може робити фото та відео з камери рухомої платформи;
- медіа-файли повинні зберігатися в базі даних;
- користувач може групувати медіа-файли в альбоми:
- користувач може видаляти медіа-файли;
- користувач може створювати, редагувати та видаляти описання до медіа-файлів.

2.3 Висновки до розділу

У цьому розділі були визначені технічні та функціональні вимоги до системи, що розробляється. Умови були створенні на основи вихідних даних та аналізі існуючих рішень

Найважливішими вимогами є принципи безпеки, вимоги до збереження даних, а також критерії управлінням рухомою платформою. Важливо зауважити, що технічні та функціональні вимоги не регламентують конкретних технологій реалізації, алгоритмів або деталей. Таким чином, технології, що будуть використовуватись для реалізації поставленої задачі, будуть обрані на основі визначених технічних та функціональних вимог.

3 ВИБІР ТА ОРҐРУНТУВАННЯ КОМПОНЕНТІВ ФІЗИЧНОГО РІВНЯ

Першим етапом у реалізації зі створення СВУАРП є формування прототипу апаратної частини. Цьому етапу варто приділити належну частину уваги, адже від апаратної частини залежать подальші характеристики налаштування системи. Прототип створюється для тестування та оцінки вихідних параметрів функціювання системи в реальних умовах. До складу компонентів прототипу входять:

- Мікрокомп'ютер
- WiFi-модуль
- Камера
- Двигуни
- Корпус
- Проміжні компоненти

Після того, як було сформовано вимоги до СВУАРП, необхідно вибрати апаратну базу для реалізації системи. Для цього потрібно розглянути основні способи і технології, розібрати їх переваги, недоліки.

3.1 Види мікрокомп'ютерів та їх призначення

Мікрокомп'ютер – самодостатній комп'ютер, зібраний на одній друкованій платі, на якій встановлені мікропроцесор, оперативна пам'ять, системи введення-виведення і інші модулі, необхідні для функціонування комп'ютера. Одноплатні комп'ютери виготовляються в якості демонстраційних систем, систем для розробників або освіти, або для використання в ролі промислових або вбудованих комп'ютерів[1]. На відміну від традиційних персональних комп'ютерів, одноплатні комп'ютери часто не вимагають установки додаткових периферійних плат[1].

Мікрокомп'ютери призначені для вбудованих систем, на відміну від мікропроцесорів, які використовуються в персональних комп'ютерах або інших застосунках загального призначення, що складаються з різних дискретних чіпів. Мікрокомп'ютери використовуються в автоматизованих пристроях, таких як автомобільні системи управління двигунами, як частини, що імплантуються в медичні пристрої, пульти дистанційного керування, офісні машини, електроприлади, електроінструменти, іграшки та інші вбудовані системи.

За рахунок зменшення розміру і вартості в порівнянні з конструкцією, яка використовує такі окремі пристрої, як мікропроцесор, пам'ять і пристрої введення / виводу, мікрокомп'ютери дозволяють більш економно управляти ще більшою кількістю пристроїв і процесів. Таким чином, мікрокомп'ютери забезпечують низьке енергоспоживання, достатню надійність системи, а також дуже гнучкість налаштування логіки у процесі програмування. На малюнку 3.1 представлений один з популярних мікрокомп'ютерів [13].

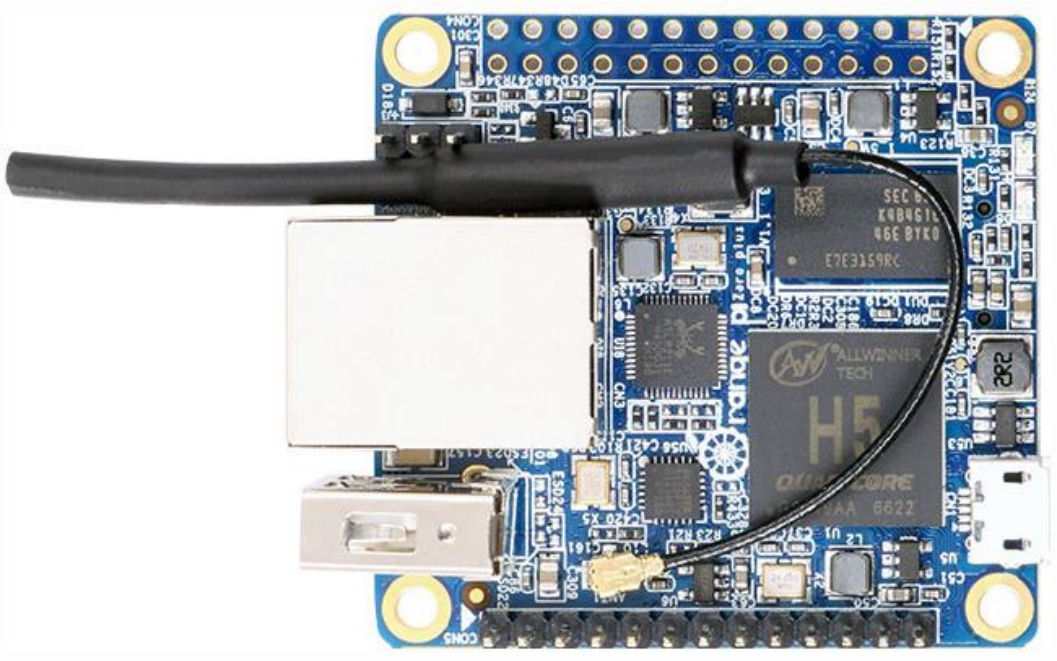

Рисунок 3.1 – Мікрокомп'ютер Orange Pi Zero Plus 512Mb

Хоча мікроконтролери і мають в більшості випадків низьку вартість, але це компенсується складністю розробки і створення прототипу для непромислових та

приватних проектів, а також відсутністю універсальності в деяких випадках.

Але поява таких рішень, як, наприклад, Arduino і одноплатних комп'ютерів вивело домашню та непромислову автоматизацію на новий рівень. Істотно знизилася вартість даних систем без шкоди надійності. Через масове впровадження в повсякденне життя так само знизилася і вартість датчиків (сенсорів) для них.

Мікрокомп'ютери з користувальницьким програмним забезпеченням надають можливість для прив'язки складних модулів до складних індивідуальних систем. Сімейство мікроконтролерів STM32, засноване на ядрі ARM CortexM3, забезпечує основу для побудови широкого спектру вбудованих систем від простих батарейних систем до складних систем реального часу, таких як автопілоти вертольотів. Ця сім'я компонентів включає десятки різних конфігурацій, що забезпечують широкий вибір розмірів пам'яті, доступних периферійних пристроїв, продуктивності та потужності.

Компоненти є досить недорогими - кілька доларів для найменш складних пристроїв, що виправдовує їх використання для більшості програм з малими обсягами. Справді, порівняємо за вартістю низькопробні компоненти «Value Line» з частинами ATmega, які використовуються для популярних плат Arduino для розробки, але мають значно більшу продуктивність і більш потужні периферійні пристрої. Крім того, використовувані периферійні пристрої є спільними для багатьох членів сім'ї (наприклад, модулі USART є спільними компонентами для STM32F1) і підтримуються однією бібліотекою програмного забезпечення.

3.1.1 Аналіз технічних характеристик Arduino

На теперішній момент доволі часто зустрічається апаратна реалізація з використання плати Arduino, але вона не може бути використана для створення апаратної інфраструктури СВУАРП. Вона реалізована на 8-ми розрядному мікроконтролері, що не має потужності обробляти потоки даних не тільки в режимі реального часу, а і принципово. Arduino – це платформа з відкритим

вихідним кодом, яка використовується для створення проектів електроніки. Arduino складається з фізичної програмованої монтажної плати (часто званої мікроконтролером) і програмного забезпечення, або IDE (Integrated Development Environment), яка працює на вашому комп'ютері і використовується для запису і завантаження комп'ютерного коду на фізичну плату[14].

Платформа Arduino стала досить популярною серед людей, які тільки починають працювати з електронікою. На відміну від більшості попередніх програмованих друкованих плат, Arduino не потребує окремого апаратного забезпечення для завантаження нового коду на плату - ви можете просто використовувати USB-кабель. На рисунку 3.2 представлена одна з моделей Arduino [14].

Рисунок 3.2 – Arduino UNO

Крім того, в середовищі IDE Arduino (рисунок 3.3) використовується спрощена версія C ++, що спрощує навчання розробки під цю плату.

Апаратне та програмне забезпечення Arduino було розроблено для художників, дизайнерів, любителів, хакерів, новачків та всіх, хто зацікавлений в створенні інтерактивних об'єктів або середовищ. Arduino може взаємодіяти з кнопками, світлодіодами, моторами, гучномовцями, пристроями GPS, камерами, інтернетом і навіть вашим смартфоном або телевізором.

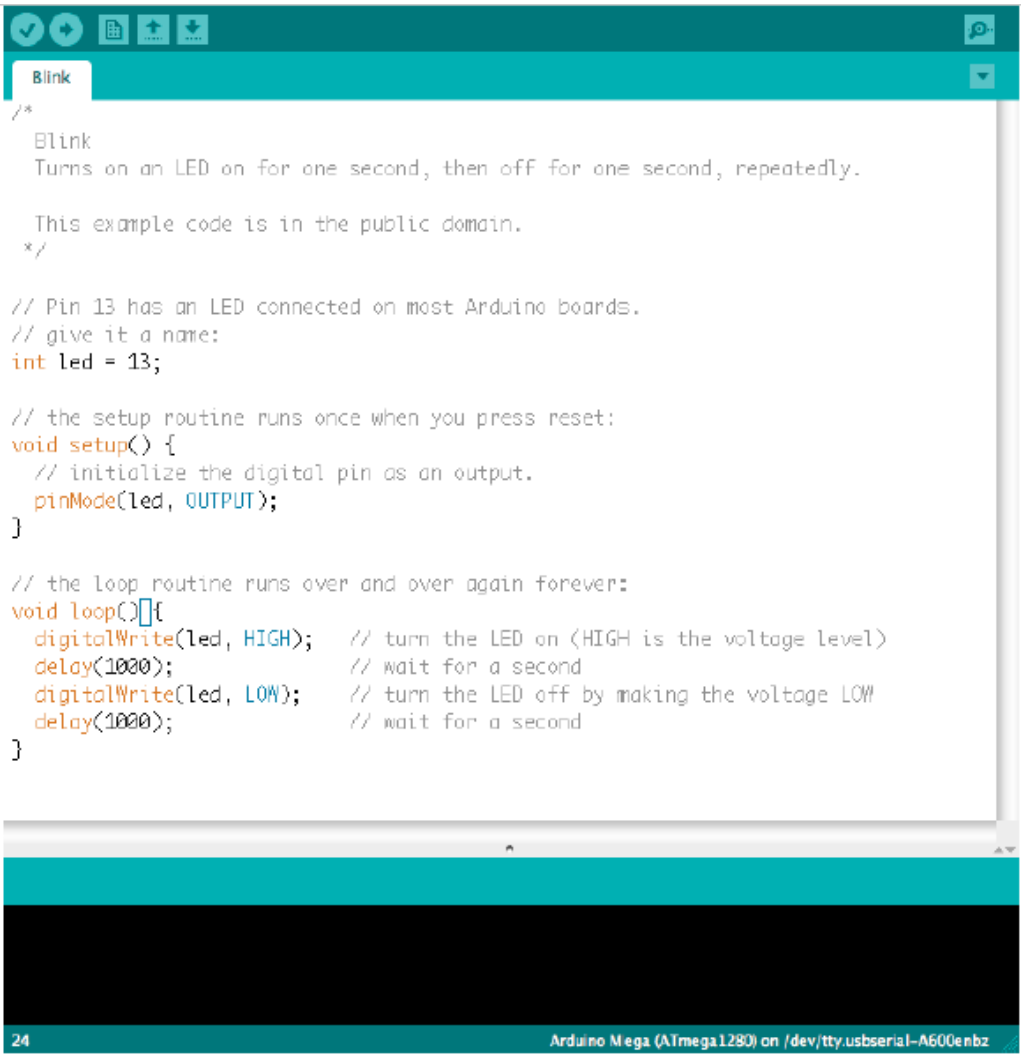

Рисунок 3.3 – Інтерфейс Arduino IDE

Ця гнучкість в поєднанні з тим, що програмне забезпечення Arduino безкоштовне, апаратні плати досить дешеві, а програмне і апаратне забезпечення легко освоїти, призвело до великої спільноти користувачів, які написали код і випустили інструкції для величезного розмаїття проектів на базі Arduino. Для всього, що завгодно: від роботів до розумних гаджетів, Arduino можна використовувати як «мозок» практично будь-якого проекту електроніки.

Платформа Arduino пропонує просту бібліотеку і єдиний ланцюг інструментів, доступний для відносно недосвідчених програмістів. Для багатьох простих систем це забезпечує швидкий шлях до прототипу. Проте простота має свої власні витрати - програмна платформа Arduino не дуже добре підходить для паралельного керування в складній системі реального часу і для програмного

забезпечення, що взаємодіє з зовнішніми пристроями, які залежать від бібліотек, розроблених за межами моделі програмування Arduino за допомогою інструментів і методів, аналогічних тим, які потрібні для STM32.

Крім того, платформа Arduino не надає можливості для режиму налагодження, що суттєво обмежує розвиток більш складних систем. Знову ж таки, режим налагодження(дебагу) вимагає виходу за межі платформи Arduino. Нарешті, середовище Arduino не підтримує операційну систему реального часу (ОСРЧ), яке є важливим при побудові більш складних вбудованих систем.

Таким чином, можна виділити наступні переваги використання Arduino:

- готовність до використання: готова до використання структура робить її дуже простою у використанні; не потрібно турбуватися про програматор, налагодження запобіжників, програмних засобів серійного монітора тощо;
- приклади коду: багато стандартних бібліотек, які полегшують і прискорюють розробку;
- легкі функції: в програмному забезпеченні Arduino є багато функцій, які роблять кодування настільки простим і швидким, яке неможливе при використанні простого мікроконтролера;
- велике співтовариство: є багато форумів в Інтернеті, на яких люди говорять про Arduino; сам сайт Arduino пояснює всі функції Arduino;
- підтримка IoT-концепції.

Також варто виділити недоліки цієї платформи:

- недостатня потужність базового мікроконтролера 8 розрядів;
- неможливість використання повноцінної POSIX-сумісної операційної системи;
- неможливість використання повноцінних розповсюджених мов програмування (Java, C#, C++);
- структура Arduino також є її недоліком; з великими структурами Arduino повинен бути дотриманий великий розмір друкованих плат;

– вартість; найважливіший фактор, який не можна заперечувати – це вартість; іноді для деяких рішень вигідніше організувати рішення на мікроконтролері.

3.1.2 Аналіз технічних характеристик Raspberry Pi

Останнім часом в області автоматизації та робототехніки набуваються популярності та практичного застосування одноплатні комп'ютери.

Одноплатний комп'ютер являє собою повноцінний комп'ютер, побудований на одній монтажній платі з мікропроцесором, пам'яттю, системи введення-виведення і іншими функціями, необхідними для роботи комп'ютера. Одноплатні комп'ютери були створені в якості демонстраційних систем або систем розробки, для освітніх систем або для використання в якості вбудованих комп'ютерних контролерів [17]. Багато типів домашніх комп'ютерів або переносних комп'ютерів об'єднують всі свої функції на одній друкованій платі.

На відміну від настільного персонального комп'ютера, одноплатні комп'ютерами часто не мають слотів розширення для периферійних функцій або пристроїв. Прості конструкції, наприклад, створені комп'ютерними любителями, часто використовують статичне ОЗУ і недорогі 8- або 16-розрядні процесори. Інші типи, такі як блейд-сервери, працюють аналогічно серверного комп'ютера, тільки в більш компактному форматі.

Зараз на ринку досить багато популярних моделей мікрокомп’ютерів від різних фірм. Одні з найпопулярніших Raspberry Pi, BeagleBone, Orange Pi, Banana Pi, Asus Tinker Board. Всі вони відрізняються ціною, технічними характеристиками, дизайном і т.д. В ході аналізу характеристики були орієнтовані на лідера з продажу та популярності - Raspberry Pi, який має відносно конкурентів середню ціну і продуктивність, але при цьому володіє найбільшим співтовариством і програмним забезпеченням з відкритим кодом.

Raspberry Pi – це крихітний комп'ютер розміром з кредитну карту. Плата має процесор, ОЗУ і типові апаратні порти, які можна знайти на більшості

комп'ютерів [9]. Головна оперативна система для Raspberry Pi - Raspbian, яка заснована на Debian. Хоча основною підтримуваною операційною системою є Raspbian, можуть бути встановлені інші операційні системи, такі як Ubuntu mare, Ubuntu Core, OSMC, RIS OS, Windows 10 IoT і багато іншого. До теперішнього часу існують кілька версій цього комп'ютера, які відрізняються розмірами, ціною і технічними характеристиками. Модель B мікроконтролера Raspberry Pi 3 зображена на рисунку 3.4 [17].

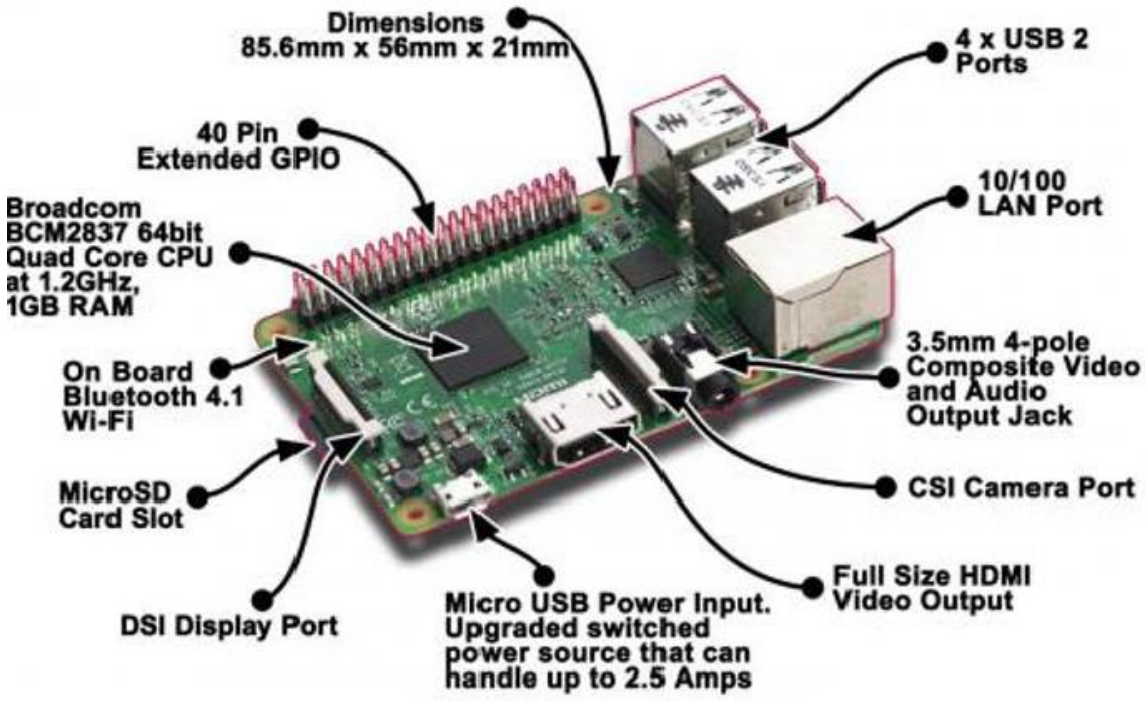

Рисунок 3.4 – Модель B мікроконтролера Raspberry Pi 3

Raspberry Pi 3 є третім поколінням мікрокомп’терів своєї моделі. Він замінив Raspberry Pi 2 Model B в лютому 2016 року. У порівнянні з моделлю Raspberry Pi 2 він отримав такі удосконалення:

– 1.2-гигабітний 64-розрядний чотирьохядерный процессор;

– ARMv8 802.11n Wireless LAN;

– Bluetooth 4.1;

– Bluetooth Low Energy (BLE).

Raspberry Pi 3 має ідентичний форм-фактор з попередніми Pi 2 (і Pi 1 Model B +) і має повну сумісність з Raspberry Pi 1 і 2. Також всі сімейство має дуже багато зовнішніх модулів, які розширюють функціонал цього комп'ютера. Наприклад, плата Gertboard (рисунок 3.5 [17]) дозволяє розширити кількість ліній вводу / виводу краще за інших можливих рішень.

Рисунок 3.5 – Gertboard плата для Raspberry Pi

Raspberry Pi 3 приблизно на 50% швидше, ніж попередник Pi 2. Він включає вбудований WIFI і Bluetooth Low Energy підключення, що робить його дійсно IOT-готовим пристроєм. Raspberry Pi 3 - це чотирьохядерний 64-розрядний процесор Arm Cortex A53 потужністю 1,2 ГГц з чіп-антеною, 4 портами USB, Ethernet, GPIO, HDMI, аудіовиходом 3,5 мм, чіпом WIFI, 1 Гб LPDDR2 для оперативної пам'яті і слот MicroSD. Карта MicroSD містить операційну систему Pi3, вона також може бути використана для зберігання файлів.

Існують також слоти для підключення периферійних пристроїв, таких як монітор, клавіатура та миша. Ці периферійні пристрої можна підключити через

будь-який з гнізд USB і HDMI. Якщо на моніторі немає HDMI, можна придбати адаптери для підключення через DVI або VGA.

Pi3 поставляється з попередньо відформатованою картою microSD об'ємом 16 ГБ. Такий об’єм пам’яті дозволяє легко встановлювати, видаляти, а потім повторно встановлювати різні дистрибутиви операційної системи Linux за допомогою простого у використанні графічного інтерфейсу. Постійною операційною системою Pi3 є Raspian, відкритий для Linux Debian OS. Порівняльна характеристика Raspberry Pi3 та NXP Pico i.MX7D представлена у таблиці 3.1.

Таблиця 3.1 Порівняльний аналіз одноплатних комп’ютерів

Характеристики	Raspberry Pi3	NXP Pico i.MX7D
Процесор	Broadcom BCM2837	NXP i.MX7D
Частота процесора, МГц	1,2 ГГц 4 ядра	1 ГГц 2 ядра
Об’єм ROM-пам’яті, Гб	Micro SD	4
Об’єм RAM-пам’яті	1 Гб	512 Мб
Мережа	Ethernet, Wi-Fi, Bluetooth	Ethernet, Wi-Fi, Bluetooth
GPIO, кількість	40	40

Найбільша зміна, яка була введена в роботу нової версії Raspberry Pi 3, є оновленням до процесора нового покоління та покращеного підключення з Bluetooth Low Energy (BLE) і BCM43143 Wi-Fi на борту. Крім того, Raspberry Pi 3 має покращене керування живленням, з оновленим джерелом живлення до 2,5 А, для підтримки більш потужних зовнішніх USB-пристроїв. Плата нового покоління зображена на рисунку 3.6.

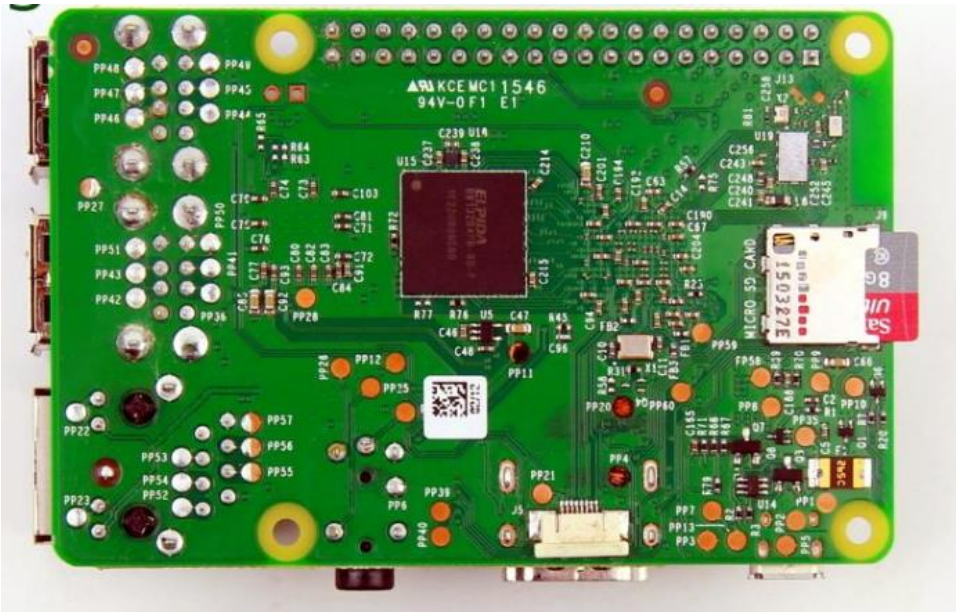

Рисунок 3.6 – Мікрокомп’ютер Raspberry Pi3

Raspberry Pi 3 має 40-контактну загальноприйняту систему вводу-виводу (GPIO), оскільки цей стандарт для пінів існує з часів моделей B + і моделі A +. Будь-яке існуюче обладнання GPIO буде працювати без змін; єдина зміна - це перемикач, на який UART піддається контактам GPIO, але це здійснюється внутрішньо операційною системою.

Ці виходи є фізичним інтерфейсом між Pi і зовнішнім світом. На найпростішому рівні, вони можуть розглядатись як перемикачі, за допомогою яких можна увімкнути або вимкнути, або які сама Raspberry Pi може увімкнути або вимкнути. Сімнадцять з 26 контактів GPIO є контактами введення/виведення; інші контакти - електричні або контакти заземлення. На рисунку 3.7 представлена візуалізація GPIO портів на платі.

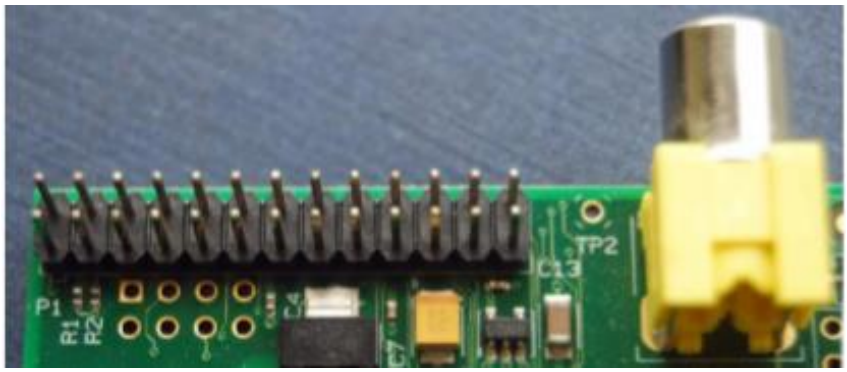

Рисунок 3.7 – GPIO на Raspberry PI B

За замовчуванням маяк приймає вхідний сигнал від підключеного пристрою через GPIO. Можна уявити собі кнопку, що передає свій статус (ввімкнено/ вимкнено) через маяк. У цій конфігурації маяк буде транслювати отримані дані в пакет телеметричної оцінки. Це визначає інформацію про двійкові стану двох контактів GPIO. Іншими словами, маяк буде регламувати два значення –0 або 1.

У вихідному режимі маяк передає дані на підключений пристрій через GPIO. Можна, наприклад, увімкнути або вимкнути світлодіодну лампу за допомогою маяка, керованого дистанційно. У цій конфігурації маяк буде доставляти дані з двох висновків щодо їх двійкових станів на підключений пристрій.

3.1.3 Обґрунтування вибору

На підставі огляду можливих рішень необхідно зробити вибір для побудови бази автоматизованої рухомої платформи. Були розглянуті найбільш поширені серед апаратних платформ Raspberry Pi і Arduino, проте відкинуто варіант з «голим» мікроконтролером тому що такий варіант є ресурсоємним за часом та складності розробки, а початкова низька вартість мікроконтролера може не компенсувати додаткових витратами на серверну та клієнтську частину майбутнього проекту.

Було проаналізовано переваги та недоліки Arduino і Raspberry Pi для організації автоматизації.

Переваги Arduino:

- архітектура з низьким енергоспоживанням (в порівнянні з Raspberry Pi);
- легко почати роботу з відмінною онлайн-підтримкою, швидке, просте створення прототипів;
- просте сполучення з датчиками і збір даних;
- дешевше, ніж Raspberry Pi (для продуктів, які не мають можливості підключення до Інтернету);

– можна відправляти дані без проводів, використовуючи Bluetooth, Rf на сервер через комп'ютер;
– безліч GPIO з можливостями PWM і дружнім до розробників;
– Повністю відкритий вихідний код.

Недоліки Arduino:

– обмеження пам'яті (вкрай мало в порівнянні з Raspberry Pi);
– для підключення до Інтернету потрібні додаткові підключені пристої.
– менш потужний у порівнянні з Raspberry Pi; не можна запустити багато важких алгоритмів, або програмувати сенсорний екран і використовуючи додаткові плати розширення; неможна запустити Open CV.

Переваги Raspberry Pi:

– високий показник продуктивності, достатня кількість пам’яті;
– ОС, заснована на Linux, а тепер навіть і Windows 10, яка може запускатися поверх неї, щоб зробити обробку більш зручною для користувача;
– велика кількість доступних GPIO – чим більше GPIO, тим більше датчиків можна з'єднати;
– дуже легко почати розробку користувачам, у яких є досвід з Linux-системами;
– мови Java, Python, C, C ++, Ruby, Go можуть бути використані для програмування – саме так, як на комп'ютері;
– користувачі успішно використовують Raspberry Pi для запуску Open CV, алгоритмів інтелектуального аналізу даних і т. д. і пов'язують результати з різними застосунками;
– з точки зору вартості, дешевше, ніж arduino у поєднанні з платою розширення Ethernet;
– широка онлайн-спільнота і нескінченні можливості того, що можна зробити з її допомогою.

Недоліки Raspberry Pi:

- потрібно добре знання Linux-систем;
- для більшості застосунків продуктивність буде надмірною, оскільки використовуватиметься в основному тільки передача даних;
- закритий вихідний код;
- за рахунок своєї потужності є більш енерговитратним.

Порівняльна характеристика плат представлена у таблиці 3.2.

Таблиця 3.2 Порівняльний аналіз плат на мікроконтролері

Характеристики	Arduino Mini	Arduino Nano	STM32F103F3
Мікроконтролер	ATmega328p	ATmega328p	STM32
Частота контролера, Мгц	16	16	16
Об'єм Flash-пам'яті, Кб	32	32	32
Об'єм RAM-пам'яті, Кб	2	2	1
Об'єм EEPROM-пам'яті, Кб	1	1	-
Кількість цифрових входів	14	14	12
Кількість аналогових входів	8	6	4

Таким чином, можна зробити висновок, що обидві платформи підходять для вибору їх за апаратну основу. Але для автоматизації з вирішенням поставлених завдань і управлінням поведінкою в реальному часі, доцільно використовувати Raspberry Pi з наступних причин:

- один з головних переваг: Raspberry Pi можна використовувати як веб-сервер, що знімає з нас завдання пошуку, налагодження та управління додатковим обладнанням; це не понесе додаткові витрати.

- крім серверної складової, для Raspberry Pi є фреймворки з відкритим вихідним кодом для організації клієнтської частини без глибоких знань в цій області (як, наприклад, WeBIOPi, який також відповідає і за серверну частину);
- доступних портів GPIO вистачить для підключення будь-яких датчиків, а для аналогових є можливість використання додаткових недорогих АЦП.
- підтримка багатозадачності;
- багато можливостей організації і з'єднання з різними веб-сервісами, які можуть виявитися корисними (SMS-оповіщення, зв'язок з twitter і ін.).

Отже, ключовим компонентом, який використовується для створення прототипу, було обрано одноплатний комп’ютер Raspberry Pi 3. Raspberry Pi 3 є новішою версією комп'ютера розміру кредитної картки з Raspberry Pi Foundation, випущеного компанією Premier Farnell.

3.1.4 Операційна система

Фонд Raspberry Pi забезпечує Raspbian, дистрибутив Linux на базі Debian для завантаження, а також сторонні Ubuntu, Windows 10 IoT Core, RISC OS і спеціалізовані медіацентри. Він пропагує Python і Scratch як основні мови програмування, з підтримкою багатьох інших мов. За замовчуванням прошивка закрита, в той час як доступний неофіційний відкритий код. Багато інших операційних систем також можуть працювати на Raspberry Pi, включаючи формально перевірене мікроядро. Інші операційні системи сторонніх виробників, доступні через офіційний веб-сайт, включають Ubuntu MATE, Windows 10 IoT Core, RISC OS і спеціалізовані дистрибутиви для медійного центру Kodi і управління класами.

Першим етапом підготовки є запис образу обраної ОС на SD-карту. Після цього потрібно вставити SD-карту в слот і харчування плати. Перша завантаження займає близько 3 хвилин, після чого комп'ютер може перезавантажитися і вам потрібно буде почекати ще одну хвилину, щоб увійти в систему. Ця затримка

відбувається тому, що система створює 128Mb аварійний SWAP і розширює SD-карту до повної ємності. У гіршому випадку завантаження (з DHCP) може зайняти до 35 секунд.

3.2 Вирішення задачі бездротової інтеграції

3.2.1 Особливості технології Wi-Fi

Для реалізації функції віддаленого управління пристроєм потрібно підключитися до локальної мережі за допомогою Wi-Fi адаптера. Wi-Fi - це сімейство радіотехнічних технологій, які зазвичай використовуються для бездротових локальних мереж (WLAN) пристроїв, які базуються на стандартах IEEE 802.11. Wi‑Fi є торговою маркою Wi-Fi Alliance, що обмежує використання терміна Wi-Fi Certified для продуктів, які успішно завершують тестування сертифікації взаємодії. Wi-Fi використовує декілька частин сімейства протоколів IEEE 802 і розроблено для безперешкодної взаємодії з протоколом Ethernet.

Сумісні пристрої можуть підключатися один до одного через Wi-Fi через точку бездротового доступу, а також до підключених пристроїв Ethernet і можуть використовувати його для доступу до Інтернету. Така точка доступу має діапазон близько 20 метрів (66 футів) у приміщенні та більший діапазон на відкритому повітрі. Покриття точки доступу може бути настільки ж невеликим, як одна кімната зі стінами, які блокують радіохвилі, або величиною з декілька квадратних кілометрів, що досягається за допомогою декількох точок доступу, що перекриваються.

3.2.2 Переваги та недоліки управління за допомогою Wi-Fi

У сучасних умовах розвитку технологій дуже широко використовується технологія Wi-Fi, проте перед її використанням у системі проаналізуємо переваги та недоліки у випадку управляння таким способом. До переваг відносять:

- більшість приладів, включаючи персональні комп’ютери, все мають вбудований модуль Wi-Fi, що робить інтеграцію максимально невитратною;
- крім управляння, надається можливість доступу до мережі, що розширяє функціональні можливості;
- технологія має підтримку серед таких великих компаній як Intel, HP, Apple, Qualcomm;
- не потребує додаткового обладнання на стороні користувача, як у випадку з радіоуправлінням, де необхідний пульт.

Серед недоліків можна виділити:

- бездротове управління займає частину ефіру у Wi-Fi-мережі, через яку ведеться управління;
- зниження швидкості передачі у мережі прямо пропорційне впливу на якість зображення і затримку передачі сигналу;
- застосування кодека зі стиснення інформації потребує чималих апаратних ресурсів.

Проаналізувавши вище наведені факти, можна зробити висновок, що використання технології Wi-Fi є доцільним і задовольняє поставлені потреби управління.

3.2.3 Обґрунтування вибору Wi-Fi модуля

Використання технології Wireless LAN є доцільним для побудови мереж, де використання кабельної системи є економічно або технічно недоцільним. Сучасні реалізації Wi-Fi дозволяють отримати швидкість передачі даних понад 100 Мбіт/с, при цьому користувачі можуть переміщуватися між точками доступу на території покриття мережі Wi-Fi, використовуючи мобільні пристрої (КПК, смартфони, PSP і ноутбуки), оснащені клієнтськими приймально-передавальними пристроями Wi-Fi та отримувати доступ в Інтернет.

Для того щоб позбутися від мережевого кабелю був використаний адаптер D-Link DWA-110, зображений на рисунку 3.10. Він має зйомну антену, а також

підтримується всіма операційними системами на базі Linux або Windows. З точки зору сумісності задовольняє поставлені вимоги, адже підтримує стандарти IEEE 802.11b, IEEE 802.11g и IEEE 802.11n.

Рисунок 3.10 – WiFi-модуль D-Link DWA-110

Для запуску драйверу на Raspberry Pi необхідно виконати наступні кроки:

– оновлено вбудоване програмне забезпечення, завантажено і встановлено файл драйвера:

sudo get update

sudo wget "https://miniboard.com.ua/index.php?controller=attachment&id_attachment=25" -O /lib/firmware/mt7601u.bin

sudo reboot

– підключено Wi-Fi до Raspberry;

– перевірено, чи визначилася вона:

#lsusb

отримано повідомлення:

Bus 001 Device 005: ID 07d1:3c07 D-Link System DWA-110 Wireless G Adapter(rev.A1) [Ralink RT2571W]

– підключено пристрій до локальної мережі:

sudo wpa_passphrase имя_точки

ключ_точки > /etc/wpa_supplicant/

wpa_supplicant.conf

sudo iwconfig wlan0 essid имя_точки

sudo wpa_supplicant -B -Dwext -i

wlan0 -c /etc/wpa_supplicant/

wpa_supplicant.conf

sudo ifconfig wlan0 down

sudo ifconfig wlan0 up

і перевірено, що пристрій підключен до точки доступу:

ifconfig

3.3 Опис характеристик камери

Важливим елементом, який буде використовуватись для організації користувацького управління, буде камера. Трансляція відео передається на екран та на основі цього користувач буде здійснювати керування траєкторією руху пристрою.

У даному прототипі була використана USB-камера. Технічні характеристики пристрою дозволяють кріпити його практично на будь-яку поверхню і в будь-якому положенні. Не дивлячись на своє пряме і головне призначення, веб камера дозволяє робити цілком якісні знімки з непоганим дозволом, сумісна з Windows- та Linux-подібними операційними системами, а також з Raspberry Pi. Камера виконана з міцного пластику стійкого до ударів, падінь і подряпин.

Камера має розширення 16 Мп, динамічне розширення 640 x 480 px. За розмірами становить 5x5 см і важить 72 г. Підтримує формат відео зі швидкість 30 кадрів за секунду(FPS), якому відповідає 1080p. Камера, використана у прототипі зображена на рисунку 3.11.

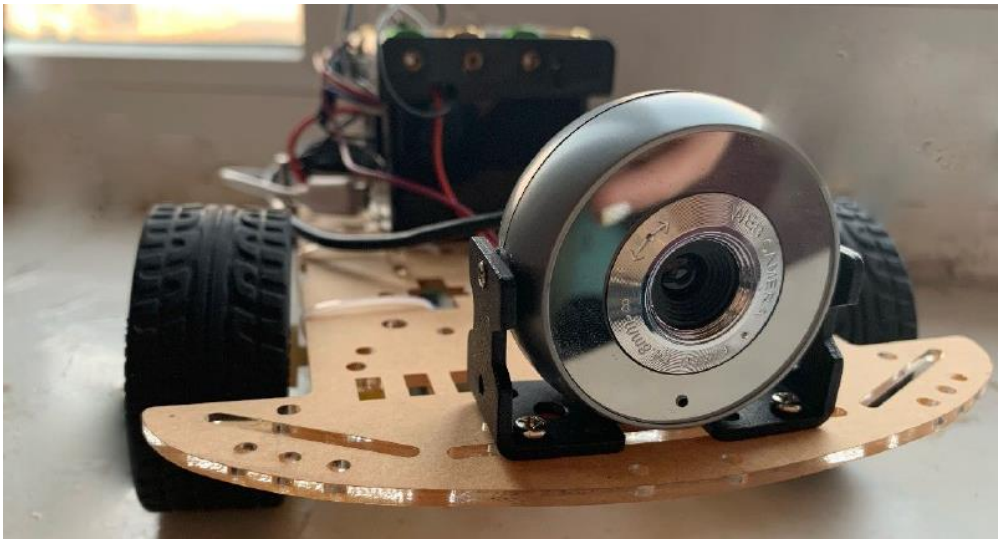

Рисунок 3.11 – USB-камера

За замовчуванням розпізнавання камери може бути вимкнено, тому для її інтеграції потрібно увімкнути функцію розпізнавання на рівні Raspberry Pi. Це можна зробити у файлі raspi-config, ввівши у командному рядку команду $ sudo raspi-config. Після цього обрати значення “Enable” (“Зробити доступною”) у пункті “Enable Camera”(“Зробити камеру доступною”). Зберегти налаштування натиснувши “Enter”.

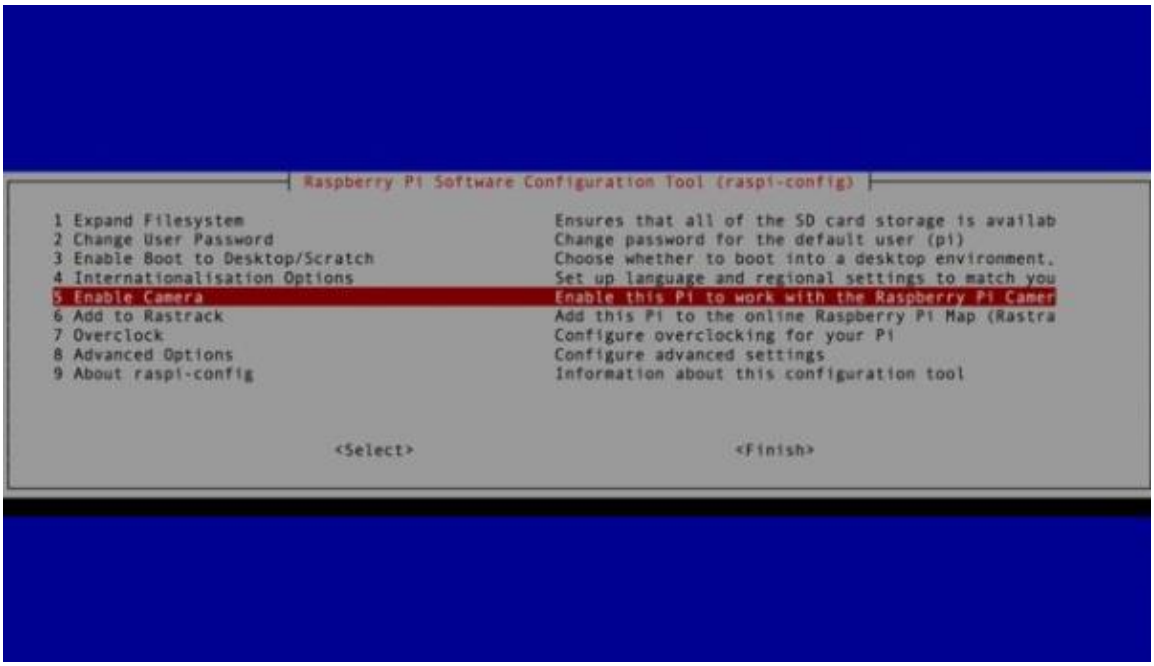

Рисунок 3.12 – Увімкнення камери через командний рядок

Для транслювання потоку відео та фото у програмі біло написано обгортку на мові програмування Java.

3.4 Двигуни

Для керування траєкторією руху платформи були використані електродвигуни. Так як до складу платформи входять два направляючих колеса, то для незалежного керування направленням їх обертання було використано два двигуна. Колеса управляються двигунами постійного струму, у яких робоча напруга від 3 до 6 вольт. Один з них зображено на рисунку 3.13.

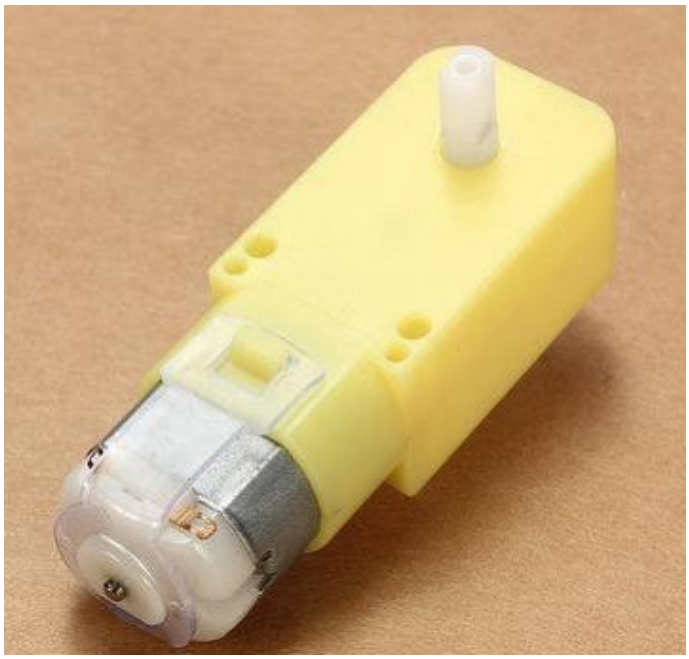

Рисунок 3.13 – Робочий двигун

Швидкість обертання колеса регулюється величиною напруги, яку ми подаємо двигуну на вхід через VCC та GND. Ці параметри будуть передаватися програмно, тому використаємо перемикач напруги. Безпосередньо підключити мотор до мікроконтролеру можна, так як типові струми пінів контролера складають кілька міліампер, а у моторів, навіть у іграшкових, рахунок може йти до десятків ампер. Теж саме з напругою:

мікроконтролер оперує напругою до 5 В, а мотори бувають різного вольтажу.

Н - міст – це електронний модуль, аналог перемикача, звичайно застосовується для живлення двигунів постійного струму і крокових двигунів, хоча для крокових двигунів зазвичай застосовуються більш спеціалізовані модулі. Позначається "Н", тому що принципова схема Н-моста нагадує букву Н. Структурна схема Н – моста зображена на рисунку 3.14.

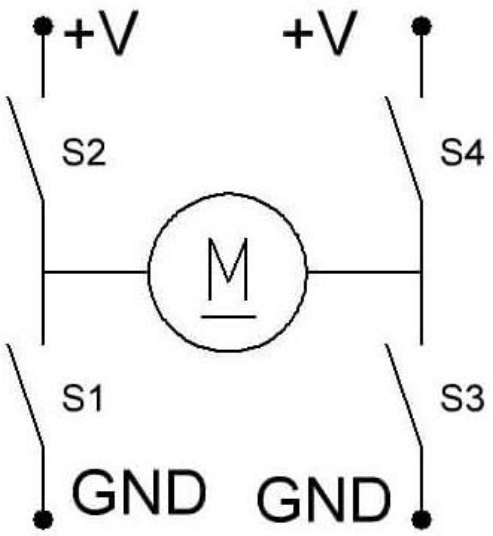

Рисунок 3.14 – Структурна схема Н – моста

Якщо замкнути контакти S1 і S4, то мотор буде обертатися в одну сторону, зліва буде нуль (S1), праворуч + напруги (S4). Якщо замкнути контакти S2 і S3, то на правому контакті мотора буде нуль (S3), а на лівому + харчування (S1), мотор буде обертатися в іншу сторону. Міст являє собою чіп L9110 з захистом від наскрізних струмів: при перемиканні контакти спочатку розмикаються, і тільки через деякий час замикаються інші контакти.

На платі встановлено два чіпа L9110 тому одна плата може управляти двома споживачами постійного струму: моторами, соленоїдами, світлодіодами, та чим завгодно, або одним двох-обмотувальним кроковим двигуном (такі крокові мотори називаються двох-фазними біполярними).

Модуль L9110 H-bridge має такі режими:

1. в режимі А - управління напрямком обертання кожного двигуна окремо;

2. в режимі B - управління швидкістю обертання двигунів за допомогою ШІМ.

Принципом роботи є те, що Н-міст служить для зміни полярності і харчування двигуна. Драйвер має два інтерфейси для підключення харчування, мікроконтролера і керованих пристроїв:

- інтерфейс для підключення керованих пристроїв складається з двох елементів, кожен з них має 2 контакту-затиску. На платі модуля ці інтерфейси позначені MOTOR A, MOTOR B;
- інтерфейс для підключення харчування і керуючих сигналів має 6 штирьових контактів. Контакти харчування модуля позначені VCC і GND. Контакти для підключення керуючих сигналів від мікроконтролера позначені A - IA, A - IB (для виходу MOTOR A); B - IA, B - IB (для виходу MOTOR B).

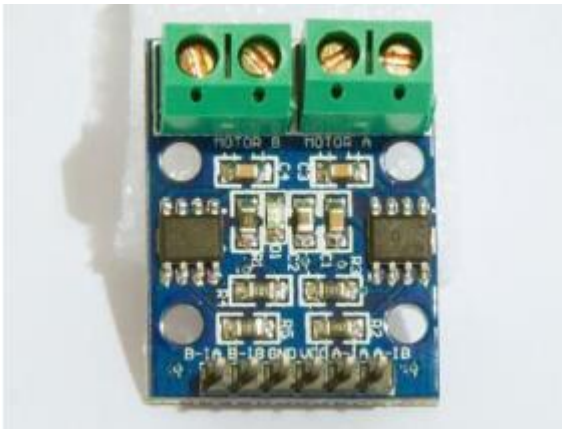

Рисунок 3.15 – Перемикач напруги L9110 H-bridge

Мотор А і Мотор В - два виходи для підключення навантаження; По-1А - сигнал «Мотор В вперед»; По-1В - сигнал «Мотор В реверс»; Земля (GND) - повинен бути з'єднаний з землею мікроконтролера і джерела живлення двигуна. Харчування (VCC) - харчування двигуна (не більше 12 В); А-1А - сигнал «Мотор А вперед»; А-1В - сигнал «Мотор А реверс». Сигнали на пінах керують напругою на виходах для підключення моторів.

Таблиця 3.3 Сигнали для двигуна A

A-1 A	HIGH	LOW
A-1 B	LOW	HIGH
Мотор A пр.	Vcc	GND
Мотор A лів.	GND	Vcc

Таблиця 3.4 Сигнали для двигуна B

B-1 A	HIGH	LOW
B-1 B	LOW	HIGH
Мотор B пр.	Vcc	GND
Мотор B лів.	GND	Vcc

Для плавного управління вихідним напругою подаємо не просто HIGH, а широтно-імпульсно модульований (PWM) сигнал. Щоб змусити мотор крутиться, потрібно на один вихід подати логічну одиницю і на другий логічний нуль. Для зміни напрямку обертання, потрібно інвертувати стан обох виходів.

3.5 Корпус

Корпус складається з щільної пластмасової пластини, яка за формою своєї поверхні передбачає кріплення множини компонентів. Корпус слугує основою, на яку будуть закріплені необхідні деталі. До складу корпусу входять колеса, виготовлені з гуми та пластику, які під’єднані до двигунів, а також частина для підпори і повороту платформи.

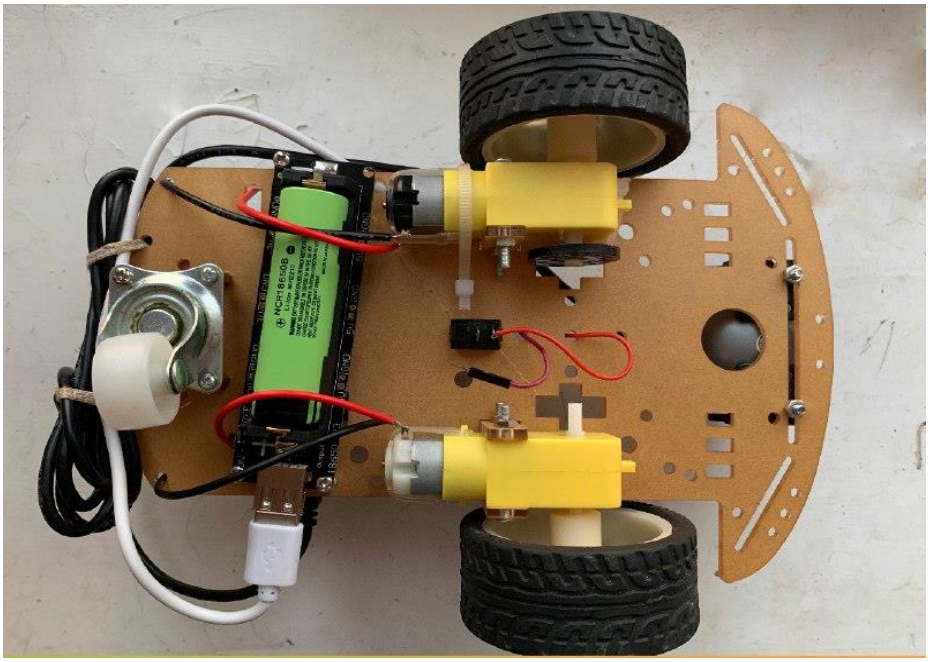

Рисунок 3.16 – Загальний вигляд корпусу

3.6 Компоненти живлення

Для приведення коліс до руху слугують двигуни, які у свою чергу потребують живлення. Для цього слугує касета, яка дає на виході напругу у 6В, до складу якої входить 4 батарейки типу АА. Касета для батарейок зображена на рисунку 3.17.

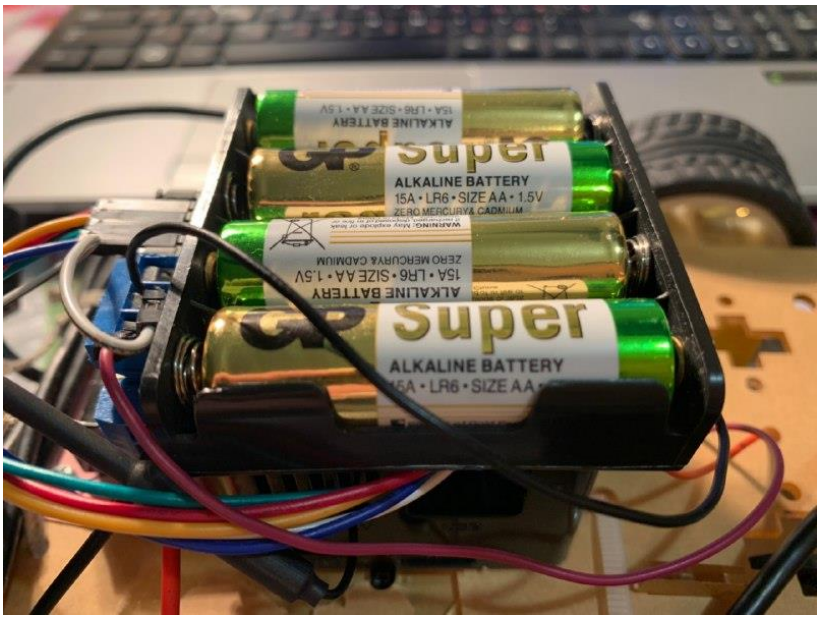

Рисунок 3.17 – Касета для батарейок

Наступним елементом, який потребує живлення, є Raspberry Pi. Для живлення мікрокомп’ютера був використаний зовнішній акумулятор NCR18650D, який дає на виході напругу 5В. Він підключається за допомогою microUSB і зображений на рисунку 3.18.

Рисунок 3.18 – Зовнішній акумулятор

3.7 Висновки до розділу

У цьому розділі були розглянуті компоненти, необхідні для реалізації фізичного рівня системи – побудови прототипу автоматизованої рухомої платформи. Саме ці компоненти забезпечують надійну, порівняно недорогу збірку прототипу, а також задовольняють потреби у реалізації задуманого функціоналу.

4 ВИБІР ТА ОРҐРУНТУВАННЯ КОМПОНЕНТІВ КЛІЄНТ-СЕРВЕРНОГО РІВНЯ

Наступним етапом у реалізації поставленої задачі є інтеграція створеного прототипу рухомої платформи з програмним забезпеченням. Для цього потрібно обрати такі засоби, які надають усі необхідні інструменти, що допоможуть у реалізації програмного продукту і дадуть змогу одержати систему, яка задовольнить користувача.

У процесі розробки були використані наступні інструменти:

- середовище розробки IntelliJ IDEA, яке має найбільші функціональні можливостей серед представлених аналогів;
- мова програмування Java для реалізації частини серверу;
- інструмент розробки Spring Boot – фреймворк, що відповідає за організацію архітектури застосунку та зв’язку усіх компонентів воєдино;
- фреймворк для автоматизації збірки проектів та підключення модулів Apache Maven;
- розподілена система керування версіями Git;
- система управління базами даних PostgreSQL, яка надає реляційний підхід;
- бібліотека Lombok, за допомогою якої скорочується кількість шаблонного коду;
- бібліотека WiringPi для доступу до GPIO-контактам з Java коду;
- бібліотека Hibernate, яка призначена для вирішення завдань об'єктно-реляційного відображення.

Технології були вибрані опираючись на критерій, що застосунок не повинен залежати від платформи та пристрою, на якому він буде розгорнутий. Саме тому були обрані такі технології, які не залежать від операційної системи.

Java була обрана як мова програмування високого рівня, серед головних переваг якої є міжплатформеність, наявність великої кількості бібліотек, які доповнюють і спрощують процес налаштування застосунку, а також

стандартизують архітектуру та стилізацію коду, що є важливим для подальшого функціонального розширення.

Системою управління базами даних була обрана PostgreSQL завдяки своїй потужності, що є важливим аспектом у продуктивності системи, особливо при великих об’ємах даних. До того ж PostgreSQL є вільним інструментом з відкритим кодом, проте яких не поступається за своєю продуктивністю таким комерційним інструментам як Oracle [6]. Для доступу до рівня бази данних та обєктно реляційного відображення було використано Spring Data JPA та Hibernate як конкретна його бібліотека реалізації. Серед головних його особливостей відмічають наявність SQL-подібної мови HQL, яка дозволяє виконувати SQL-подібні запити, записані на рівні з об'єктами даних Hibernate та зіставлення Java-класів з таблицями бази даних.

4.1 Мова програмування Java

Логіка серверної частини система написана об’єктно орієнтованою мовою високого рівня – Java. Вибір зумовлюється наявністю великої кількості фреймворків та технологій для цієї платформи, які вирішують широкий спектр прикладних задач. Ще одна перевага – міжплатформеність досягається за рахунок віртуальної машини Java (JVM).

Наступною перевагою мови є надійність роботи Java- застосунків, адже Java – об’єктно орієнтована мова високого рівня з суворою типізацією. Також мова не підтримує множинне наслідування, що унеможливлює появу виникнення «проблеми ромба», тим самим виключається ситуація неоднозначність та спрощується розуміння коду. Натомість введено поняття інтерфейс – контракт для класів-нащадків, який передбачає реалізацію методів, задекларованих у інтерфейсі. Ще одним показником надійності даної мови програмування є система обробки помилок та винятків, що розділяються на два головних види:

- виняткові ситуації, які обов’язково повинні бути оброблені програмістом, такі, що з’являються протягом роботи програми. Наприклад, користувач ввів недопустимі символи або шлях, яких прописано до файлу, не існує.
- критичні помилка під час виконання програми, пов'язана з роботою віртуальної машини Java. Наприклад, коли метод нескінченно викликає сам себе або коли недостатньо пам'яті для створення нових об'єктів[2].

Для уникнення другого типу ситуацій у Java є вбудований сервіс керування пам’яттю, який сам регулює процедуру очищення пам’яті, тому програмісту не потрібно перейматися звільненням пам’яті вручну. Тобто, розробник створює об’єкт, а вбудований збирач сміття звільнює пам'ять після того як об’єкт стає непотрібним – немає посилань на нього, отже пам'ять, виділена під цей об’єкт, буде звільнена автоматично. У Java не має підтримки вказівників, які підтримуються у мовах C та C++, наприклад. Цей факт надає можливість збирачу сміття переміщувати вказівники об’єкту з однієї області пам’яті в іншу[3].

Об’єктно-орієнтований підхід до написання коду дозволяє оперувати поняттями, що зустрічаються в реальному житті з певною долею абстракції [5]. Парадигма ООП надає такі переваги як масштабованість, що надає можливість багаторазово розширювати систему без повторного редагування і переписування коду. Масштабованість системи означає, що в систему можуть бути додані нові компоненти без впливу на все існуючі. Ще однією перевагою підтримки парадигми ООП є багаторазове використання коду, що значно скорочує його кількість та запобігає громіздкості структури системи.

На сьогоднішній день Java вважається однією з мов з найбільшим попитом у використанні, адже спектр її застосування дуже широкий – багато типів програм може бути написаним з її використанням.

4.2 Фреймворк Spring

Фреймворки та сучасні бібліотеки створенні для полегшення процесу побудови архітектури веб-застосунку та спрощення написання коду.

Програмний фреймворк (англ. software framework) –це комплекс готових до використання програмних рішень, що включають в себе архітектуру побудови проекту, логіку та базову функціональність системи або підсистеми та, навіть, дизайн [4].

Фреймворк включає в себе набір бібліотек, що містять у собі множину готових рішень, допоміжних програм, скриптів та інструментів, які спрощують процес об’єднання компонентів різного рівня архітектури програмного забезпечення. Також фреймовки використовуються для створення готового каркасу застосунку з базовими налаштуваннями. Серед головних переваг підключення фреймворків виділяють стандартизованість на структурованість застосунку.

Spring — це фреймворк з відкритим вихідним кодом, що вільно розповсюджується, створений з метою спрощення розробки корпоративних застосунків, забезпечує комплексну модель розробки і конфігурації для сучасних бізнес-застосунків на Java [5]. Ключовий елемент Spring— це підтримка інфраструктури на рівні програми: основна увага приділяється "водопроводу" бізнес-застосунків, тому розробники можуть зосередитися на бізнес-логіці без зайвих налаштувань в залежності від середовища виконання [4]. Spring можна використовувати для побудови будь-якої програми на мові Java (тобто автономних, веб-застосунків, застосунків JEE і т.д.), що позитивно відрізняє Spring від багатьох інших платформ.

Можна виділити наступні ключові концепції спрощення розробки застосунку за допомогою Spring:

- слабка зв’язаність компонентів досягається шляхом ін’єкції залежностей через сетери (англ. setter) або конструктор та використання інтерфейсів для організації взаємодії з класами. Таких підхід значно полегшує тестування системи, а також забезпечує процес взаємодії з об’єктами від третьої особи;
- інверсія управління значно полегшує процес заміни і перенесення модулів, тобто допомагає створювати незалежні один від одного компоненти і надає переваги при розробці в команді;

– поняття аспектно-орієнованого програмування і загальноприйнятих угод, які покращують читабельність коду;
– використання POJO (Plain Old Java Object) – прості Java об’єкти, на які не накладаються обмеження у наслідування чи реалізації специфічних для фреймворку класів або інтерфейсів;
– управління життєвим циклом об’єктів за допомогою контейнерів об’єктів, в ролі якого виступає Spring.

Ін’єкція залежностей (англ. Dependency Injection)—це шаблон проектування та архітектурна модель, що описує ситуацію, коли один об’єкт реалізує свій функціонал через інший об’єкт [4]. Створення необхідних залежностей для об’єкта керується спеціальним зовнішнім механізмом. Наприклад, замість явного вказання з’єднання з базою даних, можна передати його як аргумент конструктора, а все інше Spring зробить за нас. Такий механізм управління дозволяє розділити об’єкт від концепції реалізації механізмів, що надає нам перевагу у вигляді гнучкості в розробці застосунку.

Spring містить у собі незалежні один від одного модулі (рисунок 4.1 [7]). Винятком є модуль Core Container, який виступає у ролі ядра, тому все іншу модулі приєднуються до нього. Цей модуль складається з чотирьох компонентів: Beans, Core, Context, SpEL [5]

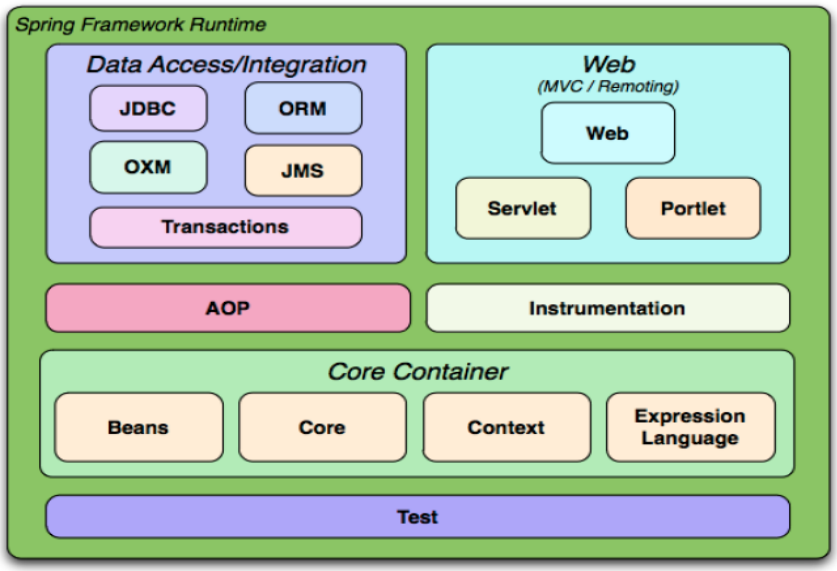

Рисунок 4.1 – Модулі Spring фреймворку

Саме компоненти Beans та Core впроваджують поняття ін’єкції залежностей та інверсію управління як найбільш фундаментальних частин фреймворку, реалізують шаблон фабрики, який заміняє необхідність у шаблоні Singleton. Фактична логіка програми відокремлена від конфігурацій та специфікацій. Підтримка інтернаціоналізації та доступ до об’єктів також реалізується інструментами складових Context.

Модуль Data Access/Integration надає інструменти для роботи з базою даних. Підмодуль ORM (англ. Object-relational mapping) надає об’єктно-реляційне відображення даним за рахунок впровадження шару, який взаємодіє з популярними бібліотеками. Наприклад, з: JPA, JDO, Hibernate та iBatus. Підмодуль JDBC (Java DataBase Connectivity), що створює шар програмної реалізації однойменного стандарту взаємодії Java застосунку з різними СУБД. Підмодуль Transaction, як зрозуміло з назви — за підтримку та обробку транзакцій [8].

Ще більш гнучке налаштування застосунку досягається за рахунок аспектно-орієнтованого програмування та модулю AOP. За контроль продуктивності застосунку відповідає модуль Instrumentation. Модульні та інтеграційні тести можуть бути написані за допомогою модуля Test.

Основний функціонал для веб-застосунків надає модуль Web. До його складу входить компонент Web-Servlet, який реалізує шаблон проектування MVC (англ. Model View Controller — модель-представлення-контролер), а також протокол HTTP. На рисунку 4.2 детально розглянуто структуру шаблону. Основою цього шаблону є чітке розділення доменної моделі, яка відокремлює бізнес логіку від представлення даних. Завдяки чітко пов’язаним даним, які забезпечує Spring, невідповідність типів буде виявлена щ на етапі валідації, тому такі помилки оброблюються програмно і не виходять на рівень системних помилок. Таким чином рекомендується пов’язувати об’єкти валідації з об’єктами бізнес логіки.

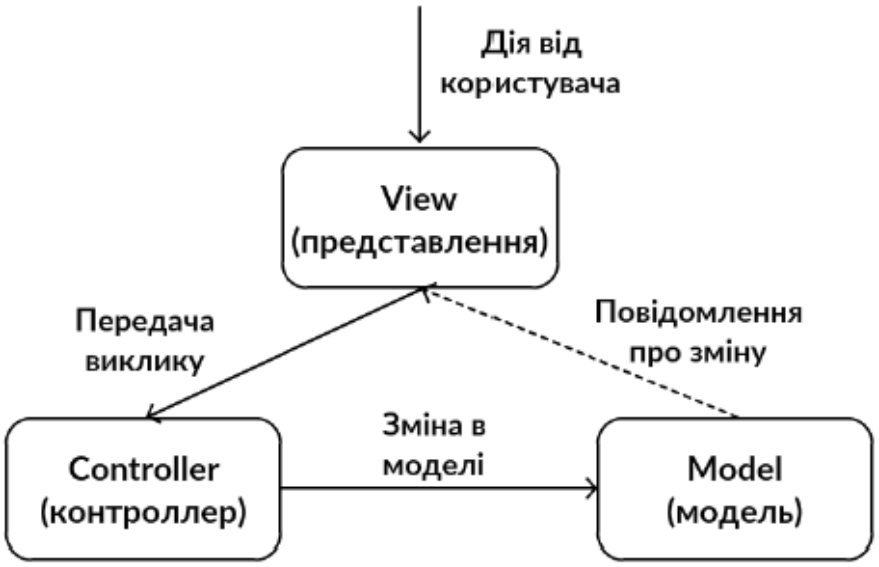

Рисунок 4.2 – Концепція шаблону MVC

До складу Spring входить центральний вбудований сервлет Dispatcher Servlet, який відповідає за розподіл запитів між контролерами. Диспетчер сервлетів – це вже відомий сервлет, який реалізує HTTP. На рисунку 4.3 представлена взаємодія диспетчера сервлетів в рамках Spring.

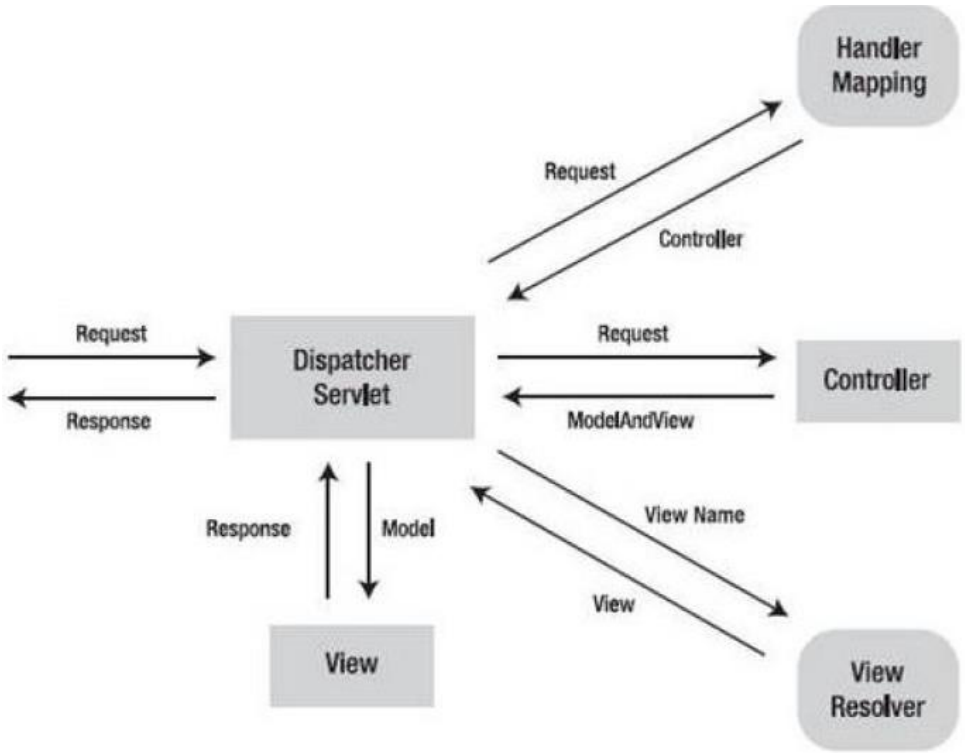

Рисунок 4.3 – Взаємодія диспетчеру сервлетів з компонентами системи

Після того як запит потрапляє до Dispatcher Servlet, він перенапрявляє його до Handler Mapping, який визначає який з контролерів обробить запит, який прийшов і повертає ім’я контролера до Dispatcher Servlet. Неможлива ситуація, коли один запит зможе обробити два контролери, адже виникає ситуація колізії і програміст отримає помилку ще а етапі розробки. Після отримання імені контролера, запит перенаправляється до цього контролера. Контролер обробляє запит і повертає об’єкт, який відповідає за відповідь на запит і відображення, яке повертається. На основі цього об’єкту відбувається пошук представлення за допомогою ViewResolver. Цей модуль дає відповідь, яка відображається на веб-частині, якщо необхідно.

Spring поміщає у контейнер класи, відмічені спеціальними анотаціями, які вказують тип даних класу, таким чином організовуючі багаторівневу архітектуру проекту. Якщо спеціально анотації над класом немає, то методи з бібліотек Spring не будуть застосовані до нього. Існує 4 основні анотації:

- @Component відмічає потребу додавання класу до контейнеру;
- @Controller відмічає клас, серед яких Dispatcher Servlet шукає метод для обробки запитів;
- @Service аналогічний до @Component, проте помічає, що цей клас передбачає обробку бізнес логіки;
- @Repository позначає клас, який взаємодіє з базою даних за допомогою запитів.

Архітектура проекту з використанням фреймворку Spring подана на рисунку 4.4.

Рисунок 4.4 – Архітектурні шари застосунку Spring

4.3 Фреймворк Spring Boot

До складу фреймворку Spring Boot входить Spring-платформа, описана в розділі 4.2, а також бібліотеки, які дозволяють запустити застосунок з мінімальними ручними конфігураціями. Spring Boot Framework забезпечує комплексну модель розробки і конфігурації для сучасних бізнес-застосунків на Java - на будь-яких платформах. Spring Boot дозволяє легко створювати автономні застосунки на базі Spring Framework, які можна "просто запустити". Він надає можливість розпочати роботу з мінімальними конфігураціями без необхідності встановлення всієї конфігурації Spring вручну. Spring Boot пропонує своїм розробникам такі переваги як підвищення продуктивність та зменшує час розробки.

Головними його перевагами вважають:

- забезпечення гнучкого способу налаштування Java Beans, XML-конфігурацій і транзакцій баз даних;
- забезпечення потужної пакетної обробки і управління REST-ендпоінтами;
- все налаштовано автоматично; не потрібні ручні конфігурації;
- підтримка класичного Spring застосунку на базі анотацій;
- полегшення управління залежностями;
- наявність вбудованого контейнеру сервлетів.

4.4 Фреймворк Spring Data JPA

Spring Data JPA являє собою модуль, який надає інструменти для роботи з шаром даних, яких за своєю структурою може бути досить громіздким. Програмісти дуже часто стикаються з шаблонним кодом для стандартних операцій і завдань, таких як вибірка даних чи пагінація. Даний модуль значно полегшує вирішення таких рутинних задач і зменшує зусилля на їх втілення за рахунок реалізації шару доступу до даних. Все що потрібно – написати інтерфейс

з методами пошуку, а Spring Data реалізує її автоматично. Модуль включає інтерфейси, які реалізовують основні методи доступу до даних.

Наслідуючи цей інтерфейс програма отримує готову реалізацію методів збереження, оновлення та видалення даних з бази, а також методи пошуку всіх елементів з таблиці за ідентифікатором [7]. Для того щоб запит до бази згенерувався автоматично, необхідно створити метод зі слів, які зможе розпізнати декомпілятор фреймворку, тобто ключові слова SQL та назви полів, задекларовані в класі-репозиторії.

.5. Фреймворк Hibernate

Hibernate є інструментом об'єктно-реляційного відображення для Java з відкритим вихідним кодом [8]. Він являє собою обгортку над такими засобами як SQL та JBDC для традиційних реляційних баз даних і є реалізацією JPA – специфікація API Java EE. Hibernate відповідає не тільки за проекцію класів Java на таблиці БД, включаючи відповідність типів даних, а й автоматично генерує запити, що скорочує витрати часу на написання SQL-коду вручну.

Проекція Java-класів на таблиці бази даних відбувається за рахунок анотацій або XML-конфігурацій. Таким чином можуть бути організовані звязки між класам один до одного, один до багатьох та багато до багатьох. Перевагою такої конфігурації є підтримка наслідування та усіх інших можливостей, доступних з Java-коду. До складу фреймворку входять наступні компоненти:

- EntityManager основний інструмент доступу до бази даних, що видаляє та створює стійкі сутності;
- EntityManagerFactory є фабрикою EntityManager відповідає за консистентність проекції даних, застосовує параметри, визначені в поточній реалізації;
- Persistence context включає множину екземплярів об’єкта, де їх життєвим цикл знаходиться під управлінням конкретним менеджером сутностей.

Немає необхідності реалізовувати специфічні інтерфейси, приналежність класу до обробки фреймворком визначається за допомогою анотацій, які декларують Hibernate правила проекцій. Під час старту застосунку анотації скануються, таким чином отримується інформація про зіставлення коду класів та структури реляційної бази даних.

4.6 Система управління базами даних PostgreSQL

В застосунку, що розробляється, необхідно постійно впорядковано зберігати набір даних, з якими працює користувач. Для цієї мети найраціональнішим рішенням буде використання бази даних. Для управління створення бази даних буде використовуватися система управління базами даних.

PostgreSQL – це вільно поширювана об'єктно-реляційна система управління базами даних (ORDBMS), найбільш розвинена з відкритих СУБД в світі і є реальною альтернативою комерційних баз даних [16]. Розробники PostgreSQL прагнуть відповідати стандарту ANSI-SQL: 2008, тому PostgreSQL відповідає вимогам ACID (атомарность, узгодженість, ізольованість і надійність) і відомий своєю вказівною і транзакційною цілісністю. Первинні ключі, що обмежують і каскадні зовнішні ключі, унікальні обмеження, обмеження NOT NULL, перевірочні обмеження та інші функції забезпечення цілісності даних дають впевненість, що тільки коректні дані будуть збережені.

У PostgreSQL безліч можливостей. Створений з використанням об'єктно-реляційної моделі, він підтримує складні структури і широкий спектр вбудованих і обумовлених користувачем типів даних. Він забезпечує розширену ємність даних і заслужив довіру дбайливим ставленням до цілісності даних. Серед головних переваг PostgreSQL виділяють:

- відкрите ПЗ відповідає стандарту SQL − PostgreSQL - безкоштовне ПЗ з відкритим вихідним кодом; дана СУБД є дуже потужною системою;
- велике співтовариство − існує досить велика спільнота, в якій можна знайти відповіді на свої питання;

- велика кількість розширень − незважаючи на величезну кількість вбудованих функцій, існує дуже багато розширень, що дозволяють розробляти дані для цієї СУБД і управляти ними;
- розширення − існує можливість розширення функціоналу за рахунок збереження своїх процедур;
- об'єктність − PostrgreSQL це не тільки реляційна СУБД, але також і об'єктно-орієнтована з підтримкою успадкування і багато іншого.

4.7 Висновки до розділу

У цьому розділі було обрано технології та алгоритми, на основі яких буде реалізована система та її архітектура. Обрані технології в повному обсязі надають можливість розробити систему, яка відповідає описаним технічним та функціональним вимогам.

Вибір мови програмування Java забезпечує міжплатформеність, а також широкий вибір бібліотек, серед яких WiringPi – бібліотека для доступу до GPIO-контактам апаратної інфраструктури.

Spring Boot Framework забезпечує комплексну модель розробки і конфігурації для сучасних бізнес-застосунків на Java, а бібліотека Hibernate надає програмну реалізацію методів взаємодії з рівнем базі даних.

Стек обраних технологій створює технічне ядро системи, на основі якого будуть реалізовані поставлені задачі.

5 РОЗРОБКА СИСТЕМИ

5.1 Процес реєстрації в системі

Для повноцінного користування розробленою системою користувач повинен авторизуватись. Якщо користувач не має облікового запису системи, він може прийняти рішення про його створення.

Якщо користувач має обліковий запис Google, він може використати його для полегшення на пришвидшення етапу реєстрації. Ім’я, прізвище та пошта будуть імпортовані з профайлу Google, а користувачу залишиться ввести лише бажаний пароль.

В іншому випадку всі поля користувач повинен буде ввести вручну, включаючи електронну адресу, яка перевіряється на наявність вже прив’язаного облікового запису до неї. Якщо етап заповнення інформації пройшов успішно, користувач отримає лист на електронну адресу про підтвердження реєстрації. Токен реєстрації буде доступним протягом 24 годин.

Після цього, цикл реєстрації закінчується і користувач може авторизуватись в системі. Діаграма діяльності процесу реєстрації наведено в додатку Б.

Для обмеженого доступу до захищених ресурсів користувача без необхідності передачі колігу та пароля використовується відкритий протокол авторизації OAuth2.

Сервер ресурсів безпосередньо зберігає захищені дані акаунтів користувачів, а авторизаційний сервер перевіряє справжність інформації, наданої користувачем, а потім створює авторизовані токени для програми, за допомогою яких додаток буде здійснювати доступ до призначених для користувача даних. Структура файлів авторизаційних класів подана на рисунку 5.1.

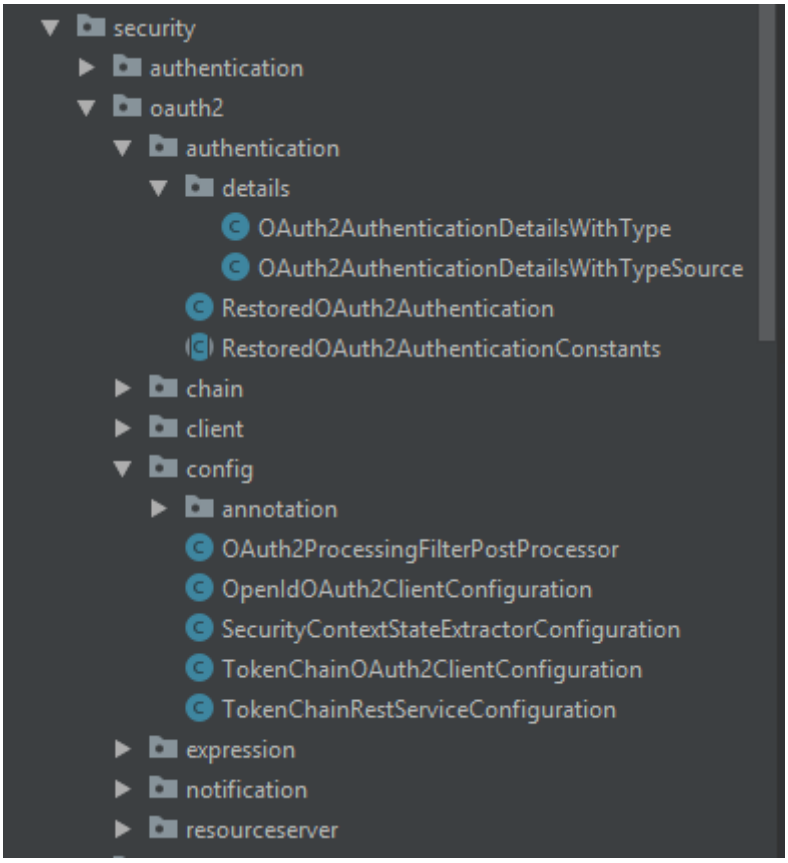

Рисунок 5.1 Структура файлів авторизаційних класів

5.2 Обробка запитів користувача

Основною функцією розроблюваної системи є обслуговування користувача за допомогою обробки дій, здійснених ним. Побудуємо та використаємо контекстну IDEF0-діаграми для детального опису системи. Першим етапом визначимо єдину роботу контекстної діаграми як «Обслуговування користувача додатку». Далі визначимо вхідні і вихідні дані, а також механізми і управління.

Для обслуговування користувача, необхідно вивести користувачеві необхідні дані, надати можливість сформувати запити, відкрити доступ до бази даних і обробити його запит. В якості вхідних даних буде використовуватися «Вихідна база даних» і «Запити користувача». Наслідком обробки запитів користувача є зміна бази даних, тому вихідними даними буде «змінена база даних». Процес буде виконуватися автоматизованими засобами програми,

використовуючи методи збереження та обробки інформації.

Основуючись на поданий опис, побудуємо контекстну діаграму роботи «Система управління автоматизованою рухомою платформою». Контекстну діаграму зображено на рисунку 5.2.

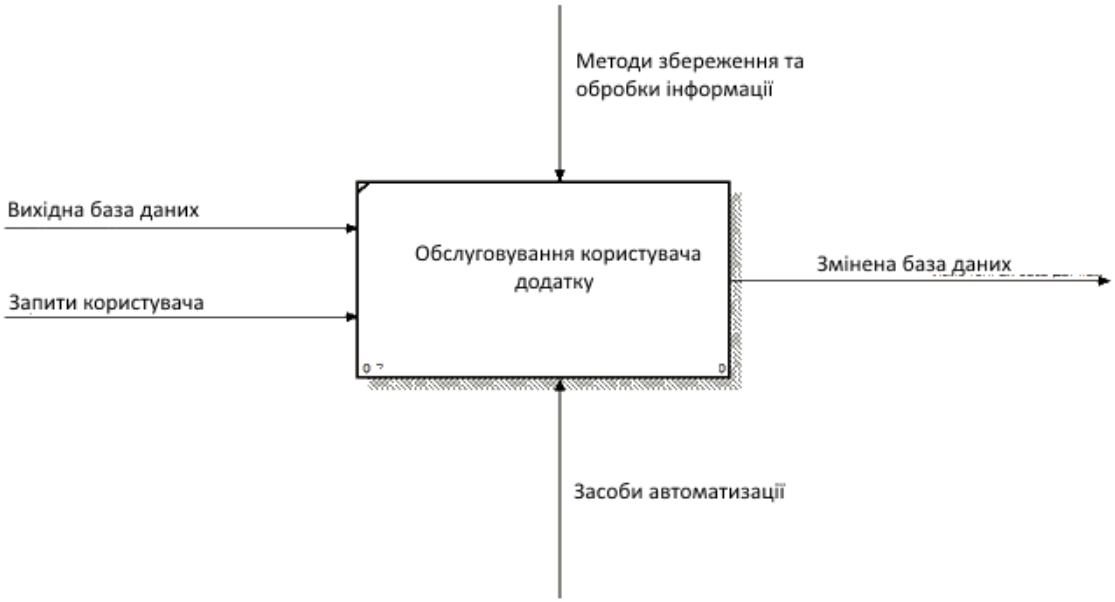

Рисунок 5.2 – Контекстна IDEF0-діаграма «Системи управління автоматизованою рухомою платформою»

Для уточнення всіх процесів зробимо декомпозицію контекстної діаграми:

– звернення до системи;
– обробка запиту користувача;
– зміна бази даних.

Побудуємо декомпозицію контекстної діаграми та деком позуємо усі наступні блоки. Результат декомпозиції контекстної діаграми зображено на рисунку 5.3.

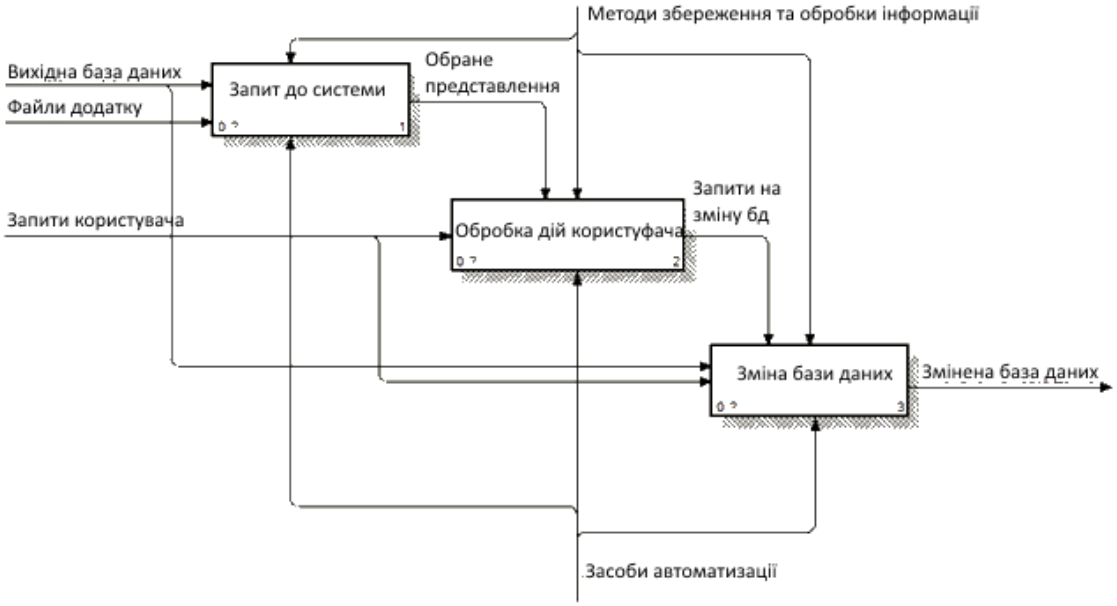

Рисунок 5.3 – Декомпозиція контекстної діаграми

5.2.1 Запит до системи

Коли відбувається запуск програми, виконується звернення до системи. Перш за все, аналізуються файли налаштувань, відмічених анотацією @Configuration. У них описуються основні компоненти програми: служби, сервіси, контент-провайдери, налаштування безпеки, оголошуються дозволи. Структура класів базової конфігурації представлена на рисунку 5.4.

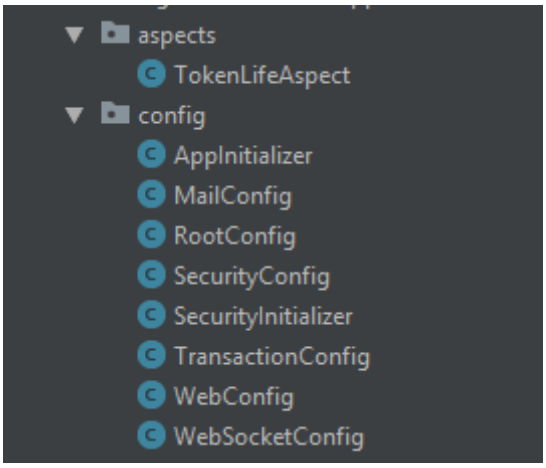

Рисунок 5.4 – Структура класів базової конфігурації

Далі, на основі вибраної операції, відбувається пошук необхідного представлення, візуального інтерфейсу. Для обраного раніше представлення завантажуються дані з бази даних, виводяться необхідні компоненти і представляються користувачеві. Результат декомпозиції роботи «Запит до системи» зображено на рисунку 5.5

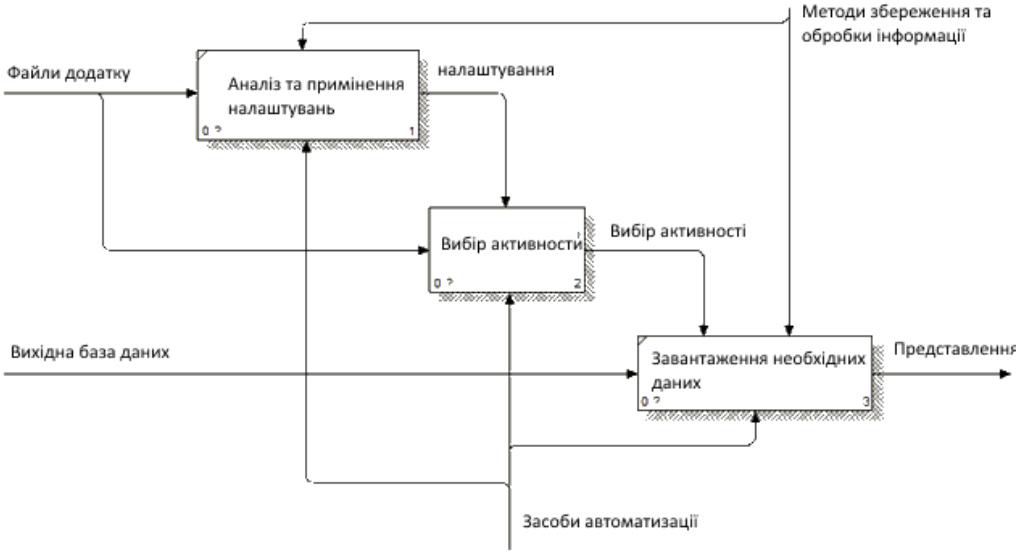

Рисунок 5.5 – Декомпозиція роботи «Запит до системи»

5.2.2 Обробка дій користувача

Після того, як користувачу представилася необхідна інформація, він отримує можливість маніпулювати цими даними. Необхідно коректно обробляти будь-які його дії.

Перш за все, система очікує введення користувачем деяких даних. Після того, як він вводить дані, і дає команду на їх обробку (наприклад, кнопка «зберегти»), система отримує дані. Далі, відбувається обробка даних, а спочатку – перевірка їх коректності. Система проводить перевірку даних і в разі некоректного введення інформує про це користувача. Якщо система отримує коректні дані – відбувається їх обробка, і формуються запити на зміну бази даних. Результат декомпозиції роботи «Обробка дій користувача» зображено на рисунку 5.6.

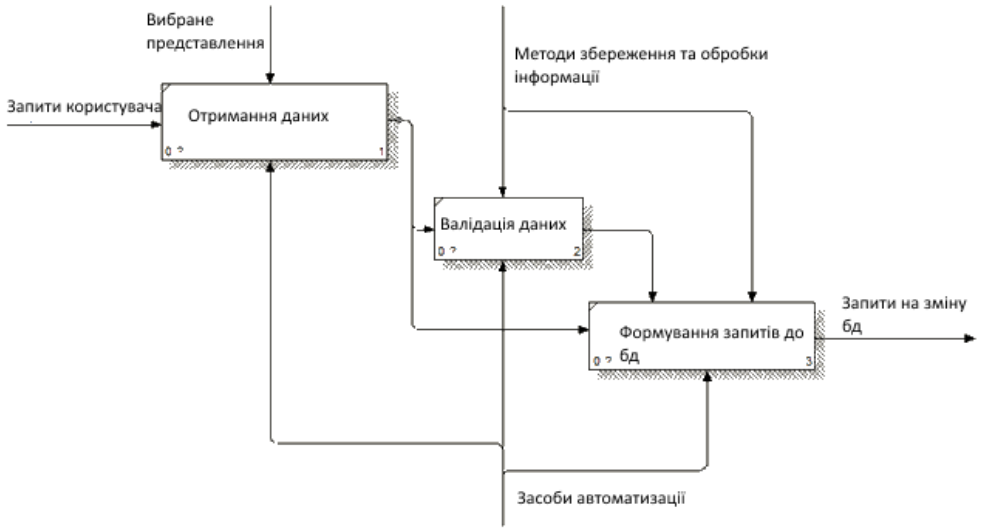

Рисунок 5.6 – Декомпозиція роботи «Обробка дій користувача»

5.2.3 Зміна бази даних

Після того як користувач відредагував певні дані, були сформовані запити на зміну бази даних. Запити необхідно обробити і внести відповідні зміни. Спочатку, в залежності від того що саме змінював користувач, вибирається необхідна таблиця. Далі, необхідно сформувати пул даних. Після формування пулу, необхідно внести його в базу даних. Результат декомпозиції роботи «Зміна бази даних» зображено на рисунку 5.7.

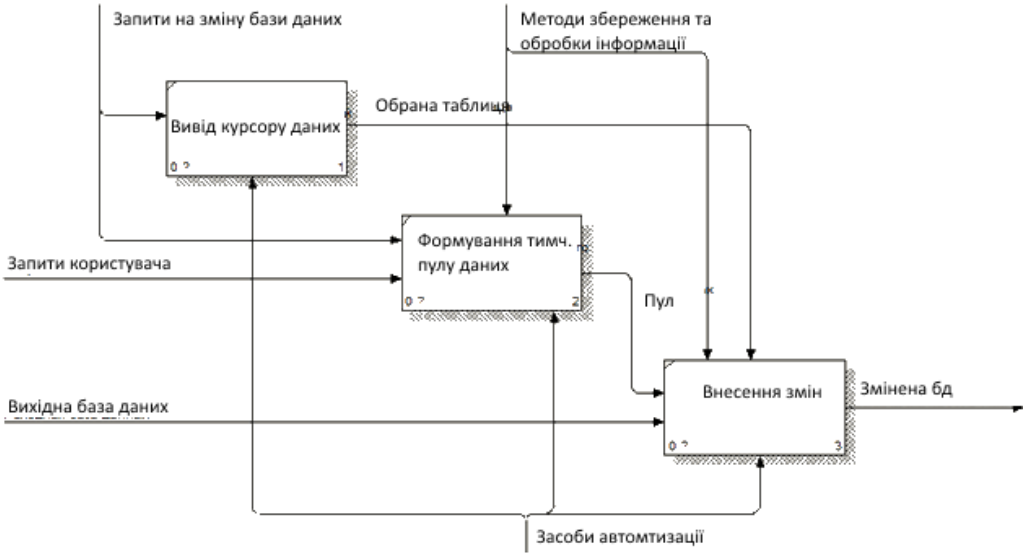

Рисунок 5.7 – Декомпозиція роботи «Зміна бази даних»

5.3 Структура бази даних

Сутності системи відображаються у таблицях бази даних. Кожна сутність має первинний ключ типу int – унікальний індекс, який застосовується для унікальної ідентифікації записів таблиці. Візуалізація структури бази даних наведена у додатку В.

Для контролю версій бази даних та застосування міграцій у разі внесення змін до структури був застосований інструмент Flyway. Він автоматично створює службову таблицю з назвою flyway_schema_history. Вона містить у собі інформацію про те, яка версія міграції застосована до бази даних з описаних у проекті. Перевірка відбувається шляхом підрахунку контрольної суми актуальної бази даних та порівняння її з останнім записом у таблиці flyway_schema_history.

Таблиця users зберігає у особі інформацію про аккаунт користувача, включаючи електронну пошту та пароль у зашифрованому вигляді алгоритмом Bcrypt. Користувач може встановити головне фото профілю, шлях до якого зберігається у колонці photo.

Таблиця verif_token використовується як тимчасове сховище введених даних користувачем під час реєстрації. Головним призначенням таблиці є збереження токена підтвердження реєстрація. Після підтвердження реєстрації дані переносяться до таблиці user, проте якщо користувач не підтвердив реєстрацію протягом 24 годин, поточний запис видаляється.

Таблиця mediafile містить у собі інформацію про створені медіа-файли користувача. Поле type може приймати два значення: photo або video, в залежності від яких відбувається фільтрація відображення об’єктів. У полі path міститься інформація про шлях знаходження файлу у файловій системі на сервері. Поле date вказує на дату створення файлу, а поле description може бути заповнене в якості примітки або описання файлу користувачем.

Таблиця folder характеризує властивості папки, такі як ім’я і опис. Поле user_id визначає користувача, який створив папку і є зовнішнім ключем, який

посилається на таблицю user.

Таблиці mediafile та folder мають зв'язок багато до багатьох, оскільки один медіа філе може знаходитись у декількох папках і в одній папці може бути декілька медіа файлів. У реляційних базах таке відношення реалізується за допомогою проміжної таблиці, яка у даному випадку має назву folder_to_mediafile. Поля сутності – це зовнішні ключі на ідентифікатори медіафайлу та папки.

База даних, яка була спроектована в ході розробки, знаходиться у третій нормальній формі. Всі атрибути є простими і множина значень, які вони можуть прийняти, містять тільки скалярні значення. Кожен неключових атрибут має функціональну залежність тільки від первинного ключа. Кожен не ключовий атрибут нетранзитивно залежить від первинного ключа [16].

5.4 Висновки до розділу

Даний розділ описує основні алгоритми взаємодії з системою, структуру проекту та його архітектуру, а також реалізацію основних функціональних можливостей.

Детально розглянуто процес реєстрації та авторизації в системі, управління рухом та траєкторією автоматизованої рухомої платформи, створення та управління медіа-файлами користувача. Використання протоколу OAuth2 забезпечує захищеність користувацьких даних, задовольняючи вимоги принципів безпеки системи. Розділення програми на модулі надало архітектурі системи чіткої структурованості.

Було досягнуто успішного результату при розробці системи, адже спроектована система задовольняє описані в розділі 2 технічні та функціональні вимоги.

6 ТЕСТУВАННЯ

У даному розділі буде розглянуто процес тестування управлінням автоматизованої рухомої платформи. Буде розглянуто основні методи його тестування та проаналізовано рівень покриття коду тестами. Також, буде проаналізовано чи тест-кейси покривають усі вимоги до застосунку та варіанти його використання.

6.1 Опис методів тестування розроблюваного застосунку

Розроблювана система є багатомодульною і включає в себе взаємодію зі сторонньою компонентою – базою даних. Тому можливі таки види тестування, як мануальне тестування, написання юніт тестів, інтеграційне тестування тощо. Основні проблеми, які повинні вирішувати тести:

- регресія – процес виникнення нових дефектів в уже тестовано коді; стійкість до регресії досягається за рахунок наявності гарантії тестами, що при внесенні помилкових змін в тестованої функціональності тести виявлять цю помилку; іншими словами, поява нового дефекту в функціональності з великою ймовірністю хоча б в одному випадку призведе до зміни між бажаним результатом і реальним;
- рефакторинг – процес зміни внутрішньої структури функціональності не зачіпаючи її зовнішню поведінку; при проведенні рефакторинга є висока ймовірність того, що у вже протестованому коді з’являться нові дефекти (регресія), до якої покриті тестами класи стійкі; у великій кількості випадків при написанні Unit тестів виникає необхідність в рефакторінгу; іншими словами з одного боку Unit тестування гарантує, що при проведенні рефакторинга ймовірність привнесення нових дефектів вкрай мала, а з іншого боку сильно пов'язаний код призводить до необхідності рефакторінга;

– дотримання контракту; контракт – формальні, точні і верифіковані вимоги до проектованого модулю; виконання функціональністю основних вимог, що висуваються до неї є необхідною умовою при проектуванні та імплементації функціональності; список вимог до функціональності досить великий і з часом, при внесенні змін тільки збільшується; тести, написані на перевірку контракту, допомагають своєчасно виявити невідповідності між контрактом якому повинна відповідати функціональність і контрактом, яким реально відповідає функціональність.

6.1.1 Модульне тестування

Модульне тестування полягає в перевірці того, як той чи інший невеликий шматок коду (модуль) обробить данні в різних ситуаціях. Тобто, наприклад, як метод об'єкта поводиться при різних вхідних даних. Саме в цьому і полягає юніт тестування в мовах, заснованих на парадигмі об'єктно-орієнтованого програмування. Модульне тестування дозволяє описати через тести вимоги до функціональності класів, однак при цьому не вирішує проблеми перевірки того, як модулі взаємодіють один з одним. А тому юніт тестування – тільки один з досить великого числа етапів тестування, через які має пройти застосунок перед тим, як потрапити до кінцевого користувача.

Для модульного тестування програмного коду, написаного на Java, існує інструмент JUnit. JUnit відноситься до сімейства програмних продуктів з відкритим вихідним кодом. Основна ідея такого тестування полягає в тому, щоб ізолювати кожен нетривіальний блок і провести верифікацію його працездатності. Всі методи цього класу, що тестуються, повинні відповідати таким вимогам:

– назва методу повинно відповідати масці - test ... (наприклад testSQLPopulation ());
– метод повинен бути public і повертати результат виконання void.

Усередині цих методів можна використовувати методи для отримання результату тестування, наприклад assertEquals, assertTrue, assertFalse і ін.

За допомогою модульних тестів були перевірені на коректність роботи класи, що відповідають за Basic та OAuth2 авторизації, а також робота застосунку паралельно на двух нодах в одному кластері. Структура класів модульних тестів представлена на рисунку 6.1.

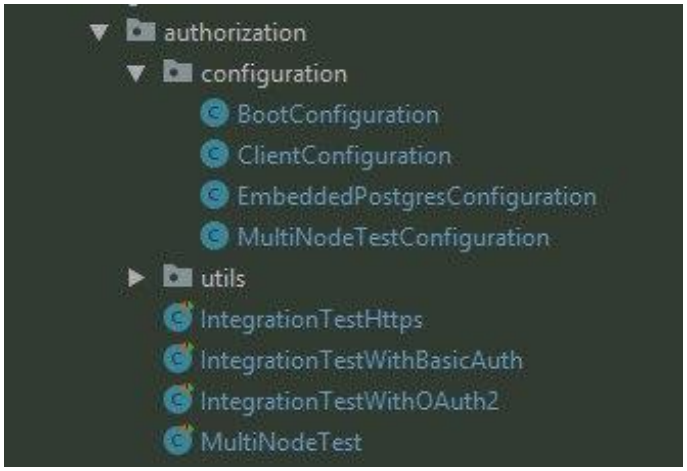

Рисунок 6.1 – Структура класів модульних тестів у Intellij Idea

Метод assertEquals порівнює два значення (наприклад, два цілих числа), перше з яких є очікуваним значенням, а друге - реальним значенням, отриманим в ході роботи програми. Метод assertNull призначений для порівняння досліджуваного об'єкта зі значенням null, тобто, фактично, для перевірки існування примірника якогось класу (не обов'язково того, який тестується, більш того, найчастіше якраз і не його). Є й інший аналогічний метод, assertNotNull, який займається схожою роботою, але вважається успішно пройденим при діаметрально протилежних умовах. Ще два методи, які порівнюють два об'єкти один з одним - assertSame і assertNotSame. Крім цих методів, є ті, які стануть в нагоді при роботі з логічними виразами, методами або змінними - це assertTrue і assertFalse. Як параметр цим методам передається булевий вираз, який потім перевіряється, відповідно, на істинність або хибність.

Результат виконання тестів наведено на рисунку 6.2, а код класів представлено у додатку Д.

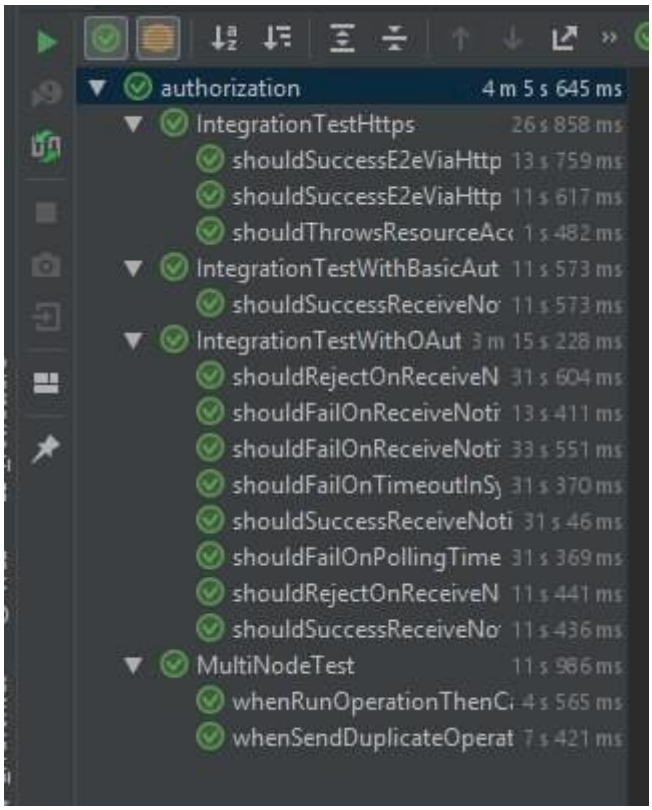

Рисунок 6.2 – Результат виконання модульних тестів у Intellij Idea

6.1.2 Інтеграційне тестування

Інтеграційне тестування - вид тестування, при якому на відповідність вимог перевіряється інтеграція модулів, їх взаємодія між собою, а також інтеграція підсистем в одну загальну систему. Для інтеграційного тестування використовуються компоненти, вже перевірені за допомогою модульного тестування, які групуються в безлічі. Дані безлічі перевіряються відповідно до плану тестування, складеним для них, а об'єднуються вони через свої інтерфейси.

Mockito - фреймворк для інтеграційного тестування на мові Java, який надає можливість проксування поведінки класів, при цьому перевіряючи саме взаємодію класів, а не конкретну поведінку класу. Структура класів інтеграційних тестів представлена на рисунку 6.3.

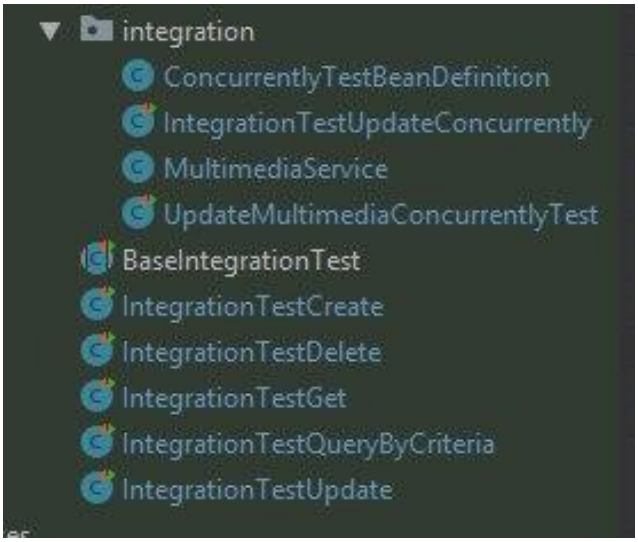

Рисунок 6.3 – Структура класів інтеграційниї тестів у Intellij Idea

Mock-Object (мок об'єкт) − це спеціально створений об'єкт, який симулює модельований клас. Він використовується в ситуаціях, коли необхідно ізолювати тестований клас від решти оточення. У термінах Mockito існує два види mock-Object:

- mock - об'єкт, виклик методів якого, якщо це не обумовлено раніше, повертає порожні значення (для примітивних типів - значення за замовчуванням, для масивів - порожні масиви);
- spy - об'єкт, виклик методів якого, якщо це не обумовлено раніше, викликає реальні методи модульованого класу.

Інтеграційним компонентом створеної системи є база даних, тому тести покривають основні операції взаємодії – створення об’єкту, видалення, модифікація, включаючи випадки, коли доступ до одно і того самого об’єкта відбувається одночасно з двох сесій. Результат виконання інтеграційних тестів наведено на рисунку 6.4.

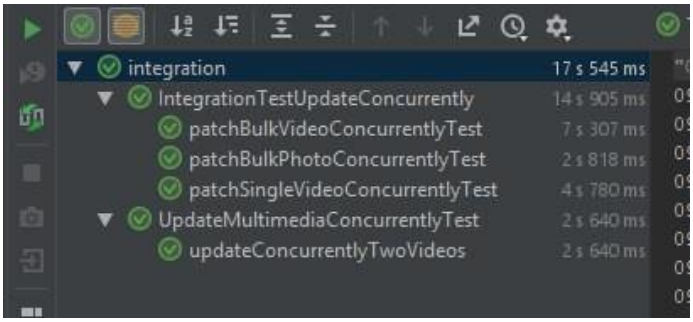

Рисунок 6.4 – Результат виконання інтеграційних тестів у Intellij Idea

6.2 Висновки до розділу

У розділі було описано процес тестування розробленої системи управління. Тестування було проведене на основі двох видів тестування: модульного та інтеграційного.

За допомогою модульного тестування була перевірена коректність роботи окремих модулів, логічно завершених функцій, які виконують певну незалежну дію. Інтеграційне тестування проводилось для того, щоб перевірити взаємодію декількох модулів, а саме взаємодію з базою даних. Протестовано також випадок паралельного доступу до даних та збереження їх консистентності.

ВИСНОВКИ

Результатом бакалаврського проекту стала запрограмована система віддаленого управління автоматизованою рухомою платформою, що відповідає усім висунутим технічних та функціональним вимогам. У ході розробки бакалаврського проекту було проаналізовано сучасні підходи до створення автоматизованих рухомих платформ та систем управління, в рамках огляду виділено основні категорії та технології їх створення.

На основі описаних технічних та функціональних вимог були визначені технології, використовуючи які буде створено проект за застосовано принципи проектування модульних систем, які покращать тестування застосунку.

Для розробки архітектури системи були застосовані сучасні шаблони проектування. Обрані принципи проектування дозволяють розширювати функціональність системи, додавати нові модулі не змінюючи існуючі. Завдяки використанню абстрактних інтерфейсів взаємодії, компоненти системи можуть бути легко замінені. Значну увагу приділено безпеці даних та надійному їх збереженню. Система була протестована за допомогою модульних та інтеграційних тестів, які у випадку зміни коду відслідкують некоректну поведінку системи.

Апаратна інфраструктура побудована на основі потужного мікрокомп’ютера Raspberry Pi 3, який за своїми характеристиками пристосований до розширення системи та підключення нових компонентів та датчиків.

Аналізуючи існуючі системи, було прийнято рішення про збереження створених користувачем медіа-файлів у базі даних PostgreSQL, що робить доступ до даних незалежним від фізичного носія – апаратної інфраструктури рухомої платформи.

Досягнуто успішного результату при розробці системи, адже спроектована система задовольняє описані в розділі 2 технічні та функціональні вимоги та є конкурентоспроможною у порівнянні з існуючими системами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Одноплатный_компьютер [Електронний ресурс] : Режим доступу: https://ru.wikipedia.org/wiki/Одноплатный_компьютер
2. Философия Java 4-ое полное издание/ Б. Эккель – Питер, 2012 – 413с.
3. Java 8 Полное руководство. Девятое издание/ Г.Шилдт – Диалектика, 2015 – 603с.
4. Spring in Action, 5th Edition/ C. Walls – MANNING, 2018 – 25с.
5. Java в облаке – Spring Boot, Spring Cloud, Cloud Founday/ Дж. Лонг, К. Бастани – Питер, 2016 – 103 с.
6. PostgreSQL: Documentation: 9.5: Index Types [Електронний ресурс]. — режим доступу: https://www.postgresql.org/docs/9.5/indexes-types.html
7. Spring Data – Modern Data Access for Enterprise Java/ T. Risberg, J. Brisbin, O. Gierke, M. Pollack, M. Hunger – O'Reilly, 2012 – 222с.
8. Java Persistence with Hibernate Second Edition/ Ch. Bauer, G. King, G. Gregory – MANNING, 2016 – 303-316с.
9. Raspberry Pi Requirements for Development Host [Електронний ресурс]: Режим доступу: https://doc.qt.io/QtForDeviceCreation/qtee-preparing-hardware-rasberrypi.html
10. WiFi Happy Cow I-Spy Mini з камерою (HC-777-270) [Електронний ресурс]: Режим доступу: https://www.yakaboo.ua/ua/tank-shpion-wifi-happy-cow-i-spy-mini-s-kameroj-hc-777-270.html#big-image-5244715
11. Himoto Barren Brushless Buggy 1:18 RTR 250 мм 4WD 2,4 ГГц (E18DBL) [Електронний ресурс] : Режим доступу: https://www.rc-hobby.com.ua/baggi/e18dbl/
12. Raspberry Pi Robots - Dexter Industries [Електронний ресурс] : Режим доступу: https://www.dexterindustries.com/raspberry-pi-robots/
13. Мини-компьютер Orange Pi Zero Plus 512Mb [Електронний ресурс] : Режим доступу: https://arduino.ua/prod2394-orange-pi-zero-plus

14. Uno Платы Ардуино [Електронний ресурс] : Режим доступу: https://doc.arduino.ua/ru/hardware/Uno
15. Структура Ардуіно [Електронний ресурс] : Режим доступу: https://apkgp.com/uk/basic.arduino
16. PostgreSQL – Википедия [Електронний ресурс] : Режим доступу: https://ru.wikipedia.org/wiki/PostgreSQL
17. Raspberry Pi Documentation [Електронний ресурс] : Режим доступу: https://www.raspberrypi.org/documentation/